US008676552B2

(12) **United States Patent**
Mech et al.

(10) **Patent No.: US 8,676,552 B2**
(45) **Date of Patent: Mar. 18, 2014**

(54) **METHODS AND APPARATUS FOR SIMULATION OF FLUID MOTION USING PROCEDURAL SHAPE GROWTH**

(75) Inventors: **Radomir Mech**, Mountain View, CA (US); **Daichi Ito**, Salinas, CA (US)

(73) Assignee: **Adobe Systems Incorporated**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 548 days.

(21) Appl. No.: **13/029,036**

(22) Filed: **Feb. 16, 2011**

(65) **Prior Publication Data**

US 2013/0132053 A1 May 23, 2013

(51) **Int. Cl.**

G06F 17/50	(2006.01)
G06F 7/60	(2006.01)
G06G 7/48	(2006.01)
G06G 7/50	(2006.01)
G09G 5/00	(2006.01)
G09G 5/02	(2006.01)
G06K 9/00	(2006.01)
G06K 9/34	(2006.01)

(52) **U.S. Cl.**
USPC **703/6**; 703/1; 703/2; 703/9; 345/582; 345/600; 382/162; 382/173

(58) **Field of Classification Search**
None
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,432,896	A	7/1995	Hwong et al.	
6,198,489	B1 *	3/2001	Salesin et al.	715/784
6,512,999	B1 *	1/2003	Dimas et al.	703/9
6,762,769	B2	7/2004	Xu et al.	
6,919,903	B2	7/2005	Freeman et al.	
6,937,275	B2	8/2005	Heiles	
6,964,025	B2	11/2005	Angiulo et al.	
6,987,535	B1	1/2006	Matsugu et al.	
7,012,624	B2	3/2006	Zhu et al.	
7,069,506	B2	6/2006	Rosenholtz et al.	
7,136,072	B2	11/2006	Ritter	
7,239,314	B2	7/2007	Johnston	
7,257,261	B2	8/2007	Suh	
7,283,140	B2	10/2007	Zhou et al.	
7,418,673	B2	8/2008	Oh	
7,430,500	B2 *	9/2008	Lei et al.	703/9

(Continued)

OTHER PUBLICATIONS

Whyte, O., Sivic, J., and Zisserman, A. 2009. Get out of my picture! internet-based inpainting in BMVC (British Machine Vision Conference), pp. 1-11.

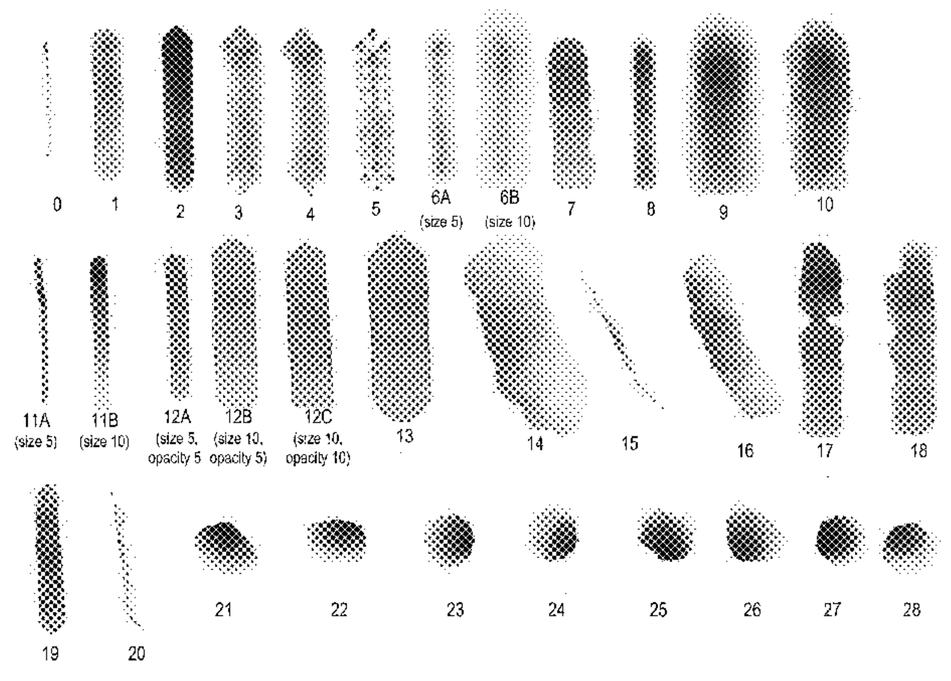
(Continued)

Primary Examiner — Omar Fernandez Rivas
Assistant Examiner — Nithya J Moll
(74) *Attorney, Agent, or Firm* — Wolfe-SBMC

(57) **ABSTRACT**

Methods and apparatus for simulating fluid motion using procedural shape growth. In a vector-based, fluid motion simulation technique, fluid location may be defined by groups of one or more polygons deposited on a digital canvas. Two or more polygons may overlap. The polygons may be semitransparent. To simulate fluid motion, vertices that specify the edges of each deposited polygon are independently moved. By moving the vertices, a polygon may grow, and fluid motion effects may be simulated, including but not limited to directional flow and blending effects. A randomization technique may be applied to the movement at each vertex to simulate the non-uniform spreading of fluids. Overlapped polygons may be blended with overlapping polygons to simulate the mixing of fluids. The technique may be applied, for example, in watercolor painting simulation, where groups of one or more polygons are deposited using brush strokes.

21 Claims, 16 Drawing Sheets



(56) **References Cited**

U.S. PATENT DOCUMENTS

7,577,313	B1	8/2009	Georgiev	
7,596,751	B2	9/2009	Rowson et al.	
7,921,003	B2 *	4/2011	Miller et al.	703/9
8,219,370	B1 *	7/2012	DiVerdi et al.	703/9
8,249,365	B1	8/2012	Winnemoeller et al.	
8,280,703	B1	10/2012	Mech	
8,335,675	B1 *	12/2012	DiVerdi et al.	703/9
2004/0218797	A1	11/2004	Ladak et al.	
2006/0267958	A1	11/2006	Kolmykov-Zotov et al.	
2012/0234722	A1	9/2012	Becvar et al.	
2013/0127890	A1	5/2013	DiVerdi et al.	
2013/0127898	A1	5/2013	DiVerdi et al.	

OTHER PUBLICATIONS

U.S. Appl. No. 12/868,519, filed Aug. 25, 2010, Elya Shechtman, et al.
U.S. Appl. No. 121857,294, filed Aug. 16, 2010, Dan Goldman, et al.
P. Bhat, B. Curless, M. Cohen, and C. L. Zitnick. Fourier analysis of the 2d screened poisson equation for gradient domain problems. In ECCV, 2008, pp. 1-14.
U.S. Appl. No. 12/394,280, filed Feb. 27, 2009, Dan Goldman, et al.
Yonatan Wexler, Eli Shechtman, Michal Irani. Space-Time Completion of Video. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 29, No. 3, Mar. 2007, pp. 1-14.
Agarwala, A., Dontcheva, M., Agrawala, M., Drucker, S., Colburn, A., Curless, B., Salesin, D., and Cohen, M. 2004. Interactive digital photomontage. In ACM SIGGRAPH, vol. 23, pp. 294-302.
Arias, P., Facciolo, G., Caselles, V., and Sapiro, G. 2011. A variational framework for exemplar-based image inpainting. IJCV 93 (July), pp. 319-347.
U.S. Appl. No. 12/315,038, filed Nov. 26, 2008, Dan Goldman, et al.
Burt, P. J., and Adelson, E. H. 1983. A multiresolution spline with application to image mosaics. ACM Trans. Graphics 2 (October), pp. 217-236.
Efros, A. A., and Leung, T. K. 1999. Texture synthesis by non-parametric sampling. IEEE Computer Society, Los Alamitos, CA, USA, pp. 1-6.
Fang, H., and Hart, J. C. 2007. Detail preserving shape deformation in image editing. In ACM SIGGRAPH, vol. 26, pp. 1-5.
Hays, J., and Efros, A. A. 2007. Scene completion using millions of photographs. In ACM SIGGRAPH, vol. 26, 4:1-4: pp. 1-7.
Kwatra, V., Essa, I., Bobick, A., and Kwatra, N. 2005. Texture optimization for example-based synthesis. In ACM SIGGRAPH, vol. 24, pp. 795-802.
Kwatra, V., Schödl, A., Essa, I., Turk, G., and Bobick, A. 2003. Graphcut textures: image and video synthesis using graph cuts. In ACM SIGGRAPH, vol. 22, pp. 277-286.
Perez, P., Gangnet, M., and Blake, A. 2003. Poisson image editing. In ACM SIGGRAPH, vol. 22, pp. 313-318.
Rother, C., Bordeaux, L., Hamadi, Y., and Blake, A. 2006. AutoCollage. In ACM SIGGRAPH, vol. 25, pp. 847-852.
Simakov, D., Caspi, Y., Shechtman, E., and Irani, M. 2008. Summarizing visual data using bidirectional similarity. In CVPR, pp. 1-8.
Szeliski, R., and Shum, H.-Y. 1997. Creating full view panoramic image mosaics and environment maps. In ACM SIGGRAPH, pp. 251-258.
Tappen, M., Freeman, W., and Adelson, E. 2005. Recovering intrinsic images from a single image. IEEE Trans. PAMI 27, Sep. 9, pp. 1459-1472.
Wei, L. Y., and Levoy, M. 2000. Fast texture synthesis using tree-structured vector quantization. In ACM SIGGRAPH, pp. 479-488.
U.S. Appl. No. 12/868,540, filed Aug. 25, 2010, Elya Shechtman, et al.
U.S. Appl. No. 13/565,552, filed Aug. 2, 2012, Elya Shechtman, et al.
Kajiya, J. T. and Kay, T. L. 1989. Rendering fur with three dimensional textures. SIGGRAPH Comput. Graph. 23, Jul. 3, 1989, pp. 271-280.
Marschner, S. R., Jensen, H. W., Cammarano, M., Worley, S., and Hanrahan, P. 2003. Light scattering from human hair fibers. In ACM SIGGRAPH 2003 Papers (San Diego, California, Jul. 27-31, 2003). SIGGRAPH '03. ACM, New York, NY, pp. 780-791.
Barnes, C., Shechtman, E., Finkelstein, A., and Goldman, D. B. 2009. PatchMatch: a randomized correspondence algorithm for structural image editing. In ACM SIGGRAPH 2009 Papers (New Orleans, Louisiana, Aug. 3-7, 2009). H. Hoppe, Ed. SIGGRAPH '09. ACM, New York, NY, pp. 1-11.
Chen, H. and Zhu, S. 2006. A Generative Sketch Model for Human Hair Analysis and Synthesis. IEEE Trans. Pattern Anal. Mach. Intell. 28, Jul. 7, 2006, pp. 1025-1040.
H. Chen and S. C. Zhu, "A generative model of human hair for hair sketching", CVPR (IEEE Conference on Computer Vision and Pattern Recognition), pp. 74-81, 2005.
David J. Heeger and James R. Bergen. Pyramid-based texture analysis/synthesis. In SIGGRAPH 95, pp. 229-238, 1995.
U.S. Appl. No. 12/858,552, filed Aug. 18, 2010.
U.S. Appl. No. 12/858,558, filed Aug. 18, 2010.
U.S. Appl. No. 12/858,546, filed Aug. 18 2010, Winnemoeller Et al.
Jonathan Duran, Deco tool and Spray Brush for creating complex, geometric patterns in Flash http://www.adobe.com/devnet/flash/articles/deco_intro.html Feb. 2, 2009, pp. 1-19.
L. Shapira, A. Shamir, and D. Cohen-Or. Image appearance exploration by model based navigation. In Computer Graphics Forum, Eurographics, 2009. (Mar. 30 to Apr. 3, 2009) pp. 1-10.
Igarashi, T. and J.F. Hughes, A Suggestive Interface for 3D Drawing, in Proceedings of the ACM Symposium on User Interface Software and Technology, UIST 01. 2001, ACM: Orlando, Florida. pp. 1-9.
J. Marks, B. Andalman, P.A. Beardsley, and H. Pfister et al. Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation. In ACM Computer Graphics (SIGGRAPH '97 Proceedings), pp. 389-400, Aug. 1997.
Sims, K. Artificial Evolution for Computer Graphics. Computer Graphics 25, Aug. 4, 1991, pp. 319-328.
U.S. Appl. No. 12/857,382, filed Aug. 16 2010, Mech Et al.
Esch G., Wonka P., Muller P., Zhang E.: Interactive procedural street modeling, SIGGRAPH 2007 Sketches.
U.S. Appl. No. 13/029,036, filed Feb. 16, 2011, Adobe Systems Incorporated.
U.S. Appl. No. 13/219,848, filed Aug. 26, 2011, Adobe Systems Incorporated.
Deussen O. and Lintermann, B.: Interactive Modeling of Plants. IEEE Computer Graphics and Applications, 19 (1999), 1, pp. 56-65.
Muller P., Wonka P., Haegler S., Ulmer A., Gool L.V.: Procedural modeling of buildings. ACM Trans. Graph., 25(2006) 3, pp. 614-623.
Wonka P., Wmmer M., Sillion F., Ribarsky W.; Instant architecture. ACM Trans. Graph.,22 (2003), 3, pp. 669-677.
Parish Y. I. H., Muller P.; Procedural modeling of cities. In Proceedings of ACM SIGGRAPH 2001, pp. 301-308.
Wong M. T., Zongker D. E., Salesin D. H.; Computer-generated floral ornament. In Proceedings of SIGGRAPH 1998, pp. 423-434.
Prusinkiewicz P., James M., Mech R.; Synthetic topiary. In Proceedings of ACM SIGGRAPH 1994, pp. 351-358.
Mech R., Prusinkiewicz P.; Visual models of plants interacting with their environment. In Proceedings of ACM SIGGRH 1996, pp. 397-410.
Muller M., Heidelberger B., Teschner M., Gross M.: Meshless deformations based on shape matching. ACM Transactions on Computer Graphics, 24, 3 (2005), pp. 471-478.
Barla P., Breslav S., Thollot J., Sillion F., Markosian L.; Stroke Pattern Analysis and Synthesis. Computer Graphics Forum, 25 (3), pp. 663-671, 2006.
Lloyd S. P.: Least Squares quantization in pcm. IEEE Trans. on Information Theory 28, 2 (1982), pp. 129-137.
Jodoin P. M., Epstein E., Granger-Piche M., Ostromoukhov V.: Hatching by example: a statistical approach. In Proceedings of NPAR 2002, pp. 29-36.
Hertzmann A., Oliver N., Curless B., Seitz S.M.: Curve analogies. In Rendering Techniques 2002: 13th Eurographics Workshop on Rendering, pp. 233-246.
Salisbury M. P., Anderson S. E., Barzel R., Salesin D.H.: Interactive pen-and-ink illustration, in Proceedings of SIGGRAPH 1994, p. 101.
Winkenbach G., Salesin D. H.: Computer generated pen-and-ink illustration. In Proceedings of SIGGRAPH 1994, pp. 91-100.

(56) **References Cited**

OTHER PUBLICATIONS

Deussen O., Hiller S., Van Overveld C., Strothotte T.: Floating points: A method for computing stipple drawings. Computer Graphics Forum 19, pp. 1-3 (2000).
Kwatra, V., Schodl, A., Essa, L, Turk, G., Bobick, A.: Graphcut textures: Image and video synthesis using graph cuts. Acm Trans. Graph. 22 (2003), 2, pp. 277-286.
Hertzmann, A., Jacobs, C.E. Oliver, N., Curless, B., Salesin, D.H.: Image Analogies. In Proceedings of SIGGRAPH 2001, pp. 327-340.
Ashikhmin, M.: Synthesizing Natural Textures. In Proceedings of the Symposium on Interactive 3D Graphics 2001, pp. 217-226.
Turk G.: Texture Synthesis on Surfaces. In Proceedings of SIGGRAPH 2001, pp. 347-354.
Wei L. Y., Levoy M.: Fast Texture Synthesis using Tree-structured Vector Quantization. In Proceedings of SIGGRAPH 2000, pp. 479-488.
Efros A. A., Leung T.K.: Texture synthesis by nonparametric sampling. In IEEE Int. Conf. on Computer Vision (1999), pp. 1033-1038.
Heeger. D. J., Bergen, J. R.: Pyramid-based texture analysis and synthesis. In Proceedings of SIGGRAPH 1995, pp. 229-238.
Ijiri T., Owada S., Igarashi T.: The sketch L-System: global control of tree modeling using free-form strokes. In Proceedings of 6th International Symposium SmartGraphics 2006, pp. 138-146.
U.S. Appl. No. 12/039,164, filed Feb. 28 2008, Radomir Mech.
Prusinkiewicz, Przemyslaw; Lindenmayer, Aristid (1990). The Algorithmic Beauty of Plants. Springer-Verlag. Chapter 4, pp. 101-107. ISBN 978-0387972978.
Chen, X., Neubert, B., Xu, Y.-Q., Deussen, O., and Kang, S. B. 2008. Sketch-based tree modeling using markov random field. ACM Trans. on Graphics 27, 5, pp. 1-9.
"ArtRage 3", *Ambient Design*, http://www.ambientdesign.com/, from web.archive.org on Aug. 23, 2011, (May 25, 2010),1 page.
"Auryn Inc. Auryn Ink", *iTunes Preview*, retrieved from <http://itunes.apple.com/us/app/auryn-Aurynl% 201nklid407668628>, (Jul. 20, 2011), 2 pages.
"Auryn Ink Review", www.giggleapps.com, (Feb. 10, 2011), 4 pages.
"Auryn Releases "Auryn Ink" Watercolor App; Named One of the Seven Most Innovative iPad Apps of 2010 by fast Company", www.marketwire.com, (Jan. 4, 2011), 2 pages.
"Corel Painter 11 Overview", retrieved from <http://web.archive.org/web/20100526115240/http://www.corel.com/servlet/Satellite/us/en/Product/1166553885783#tabview=tab0> on Aug. 23, 2011, 3 pages.
"Non-Final Office Action", U.S. Appl. No. 13/219,453, filed Sep. 12, 2013, 12 pages.
"Non-Final Office Action", U.S. Appl. No. 13/219,457, filed Oct. 15, 2013, 24 pages.
Ando, Ryoichi et al., "Vector Fluid: A Vector Graphics Depiction of Surface Flow", *NPAR '10: Proceedings of the 8th International Symposium on Non-photorealistic Animation and Rendering* (pp. 129-135). New York, NY, USA: ACM, (2010), 7 pages.
Ando, Ryoichi et al., "Vector Graphics Depicting Marbling Flow", *Computers & Graphics*, vol. 35, Issue 1, (Nov. 29, 2010), 14 pages.
Baxter, et al., "A viscous paint model for interactive applications", *Computer Animation and Virtual Worlds*, (2003), pp. 433-441.
Deussen, Oliver et al., "The Elements of Nature: Interactive and Realistic Techniques", *SIGGRAPH 2004 Course 31*, (2004), 64 pages.
Diverdi, et al., U.S. Appl. No. 13/219,457, filed Aug. 26, 2011, 104 pages.
Hang, Chu S., "Making Digital Painting Organic", *PhD Thesis*, Hong Kong University of Science and Technology, (Aug. 2007), 126 pages.
Laerhoven, et al., "Real-time Watercolor Painting on a Distributed Paper Model", *Proceedings of the Computer Graphics International*, (Jun. 2004), 4 pages.
McVeigh, Chris "ArtRage Studio Pro 3", *PCWorld*, retrieved from <http://www.pcworld.com/businesscenter/article/189221/artrage_studio_pro_3.html> on Aug. 23, 2011, (Feb. 12, 2010), 4 pages.
Van Laerhoven, Tom "An Extensible Simulation Framework Supporting Physically-based Interactive Painting", *PhD thesis*, University of Limburg, (Jun. 21, 2006), 171 pages.

* cited by examiner

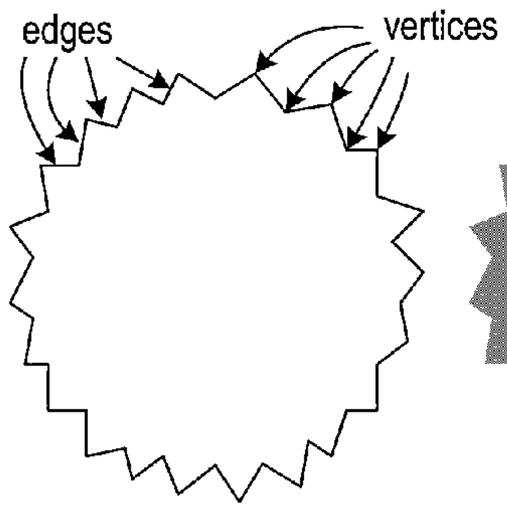


FIG. 1A

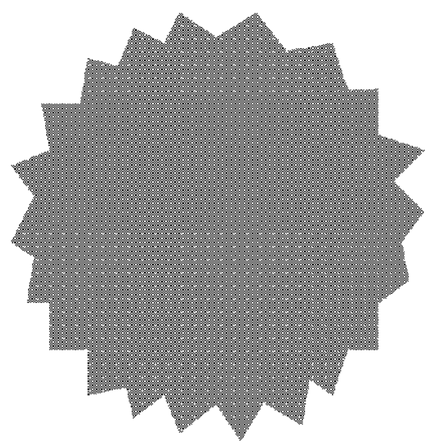


FIG. 1B

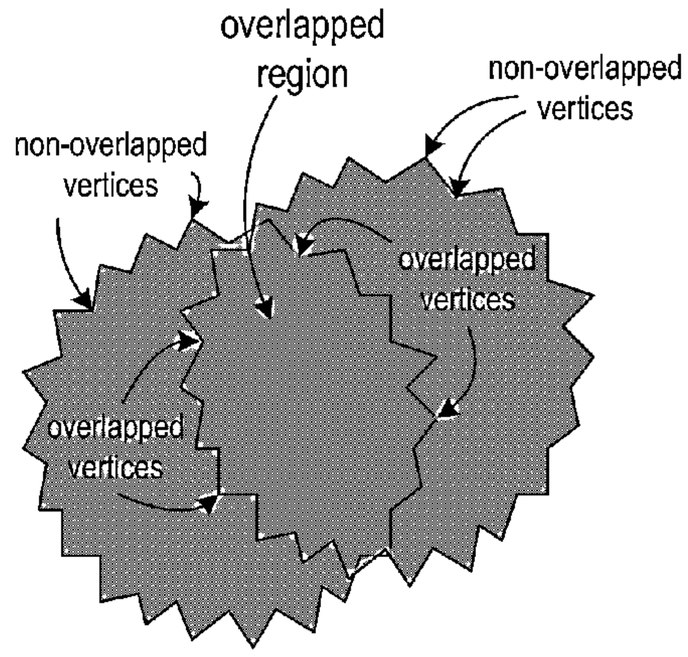


FIG. 1C

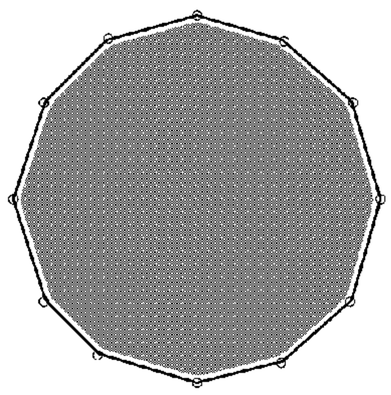


FIG. 2A

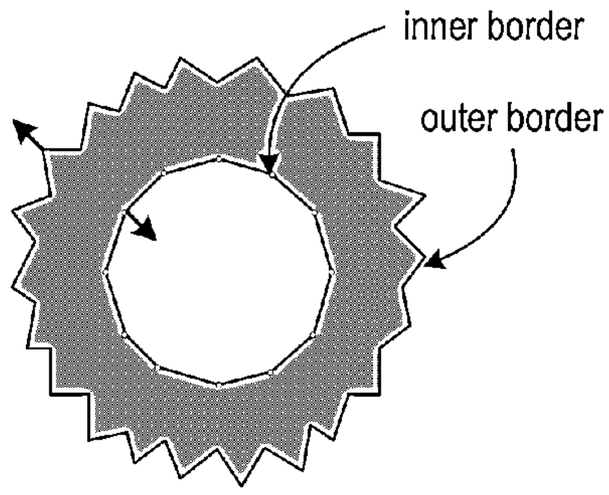


FIG. 2B

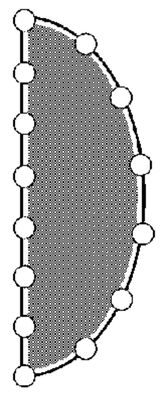


FIG. 2C

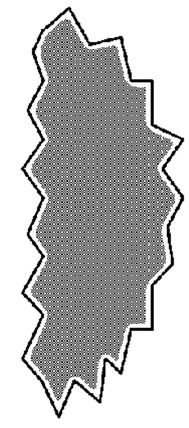


FIG. 2D

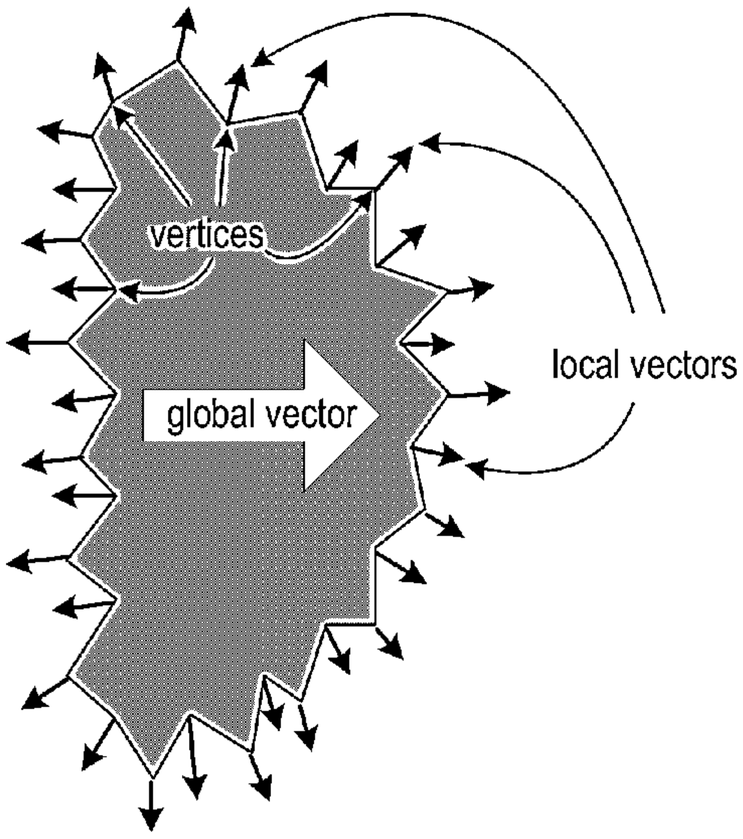


FIG. 3

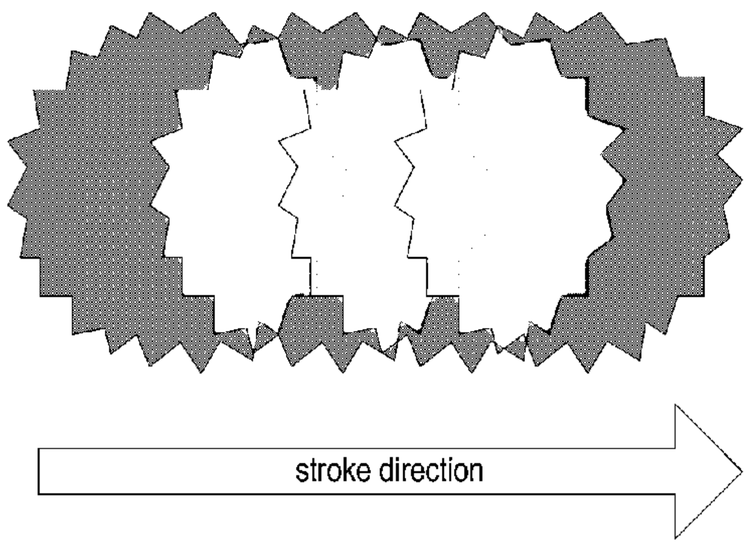


FIG. 4

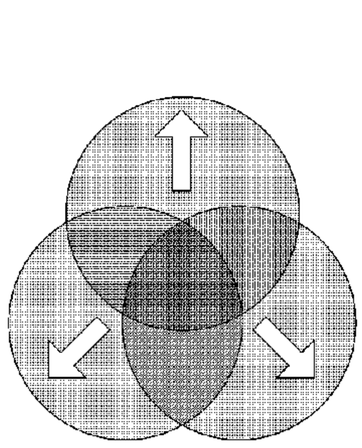


FIG. 5

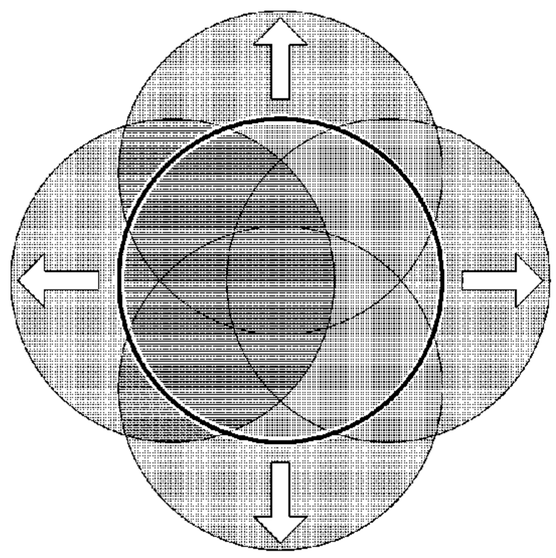


FIG. 6

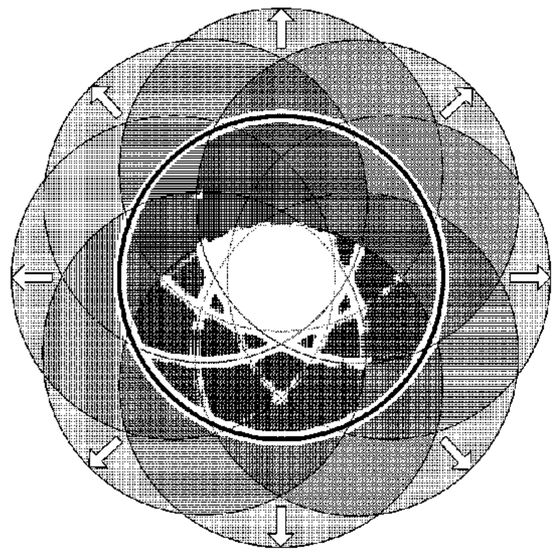


FIG. 7

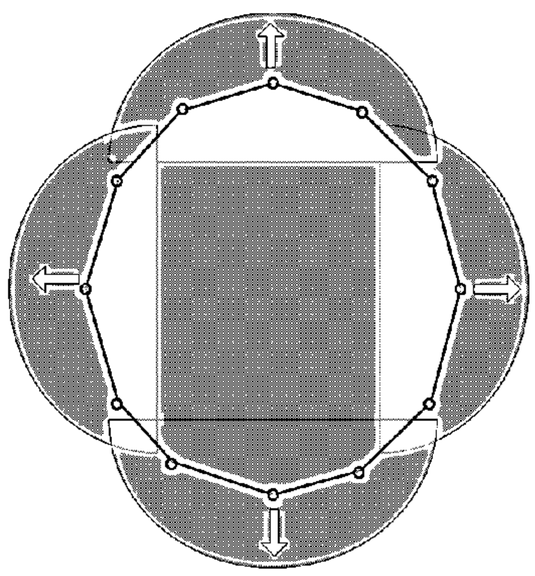


FIG. 8

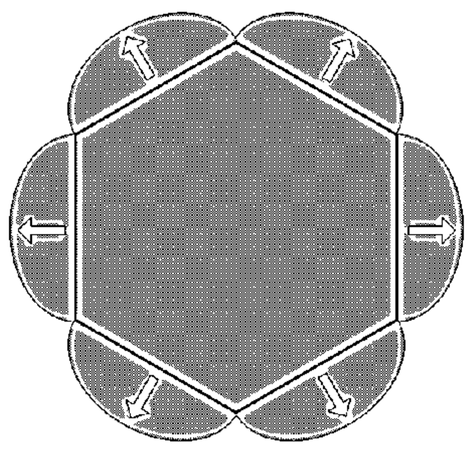


FIG. 9

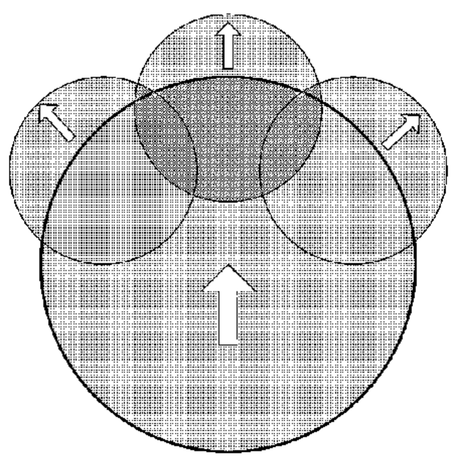


FIG. 10

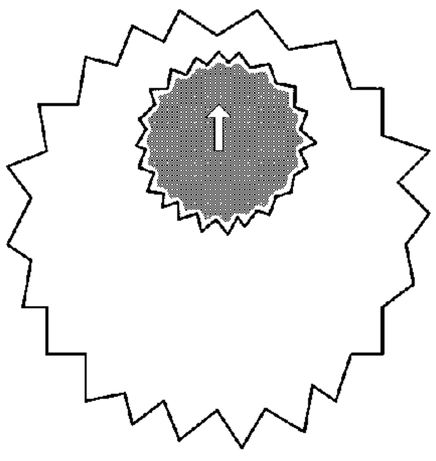


FIG. 11

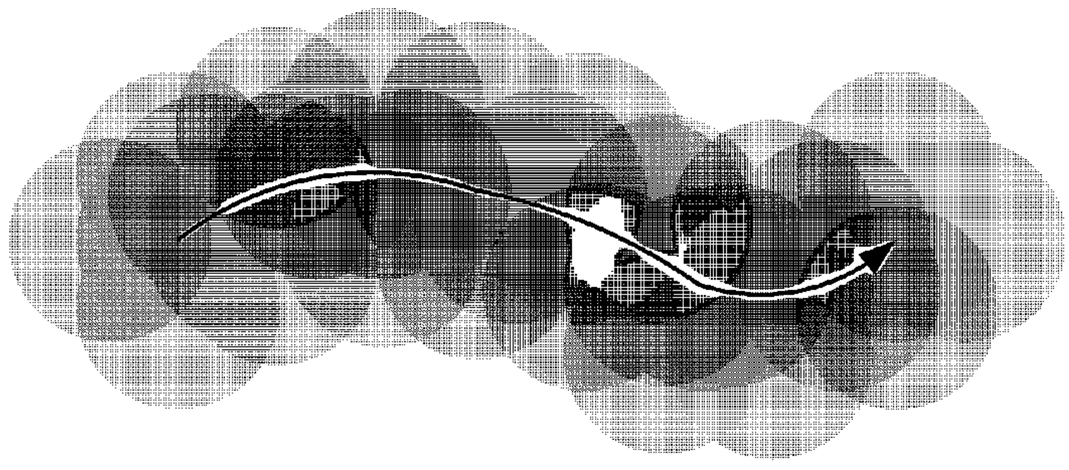


FIG. 12

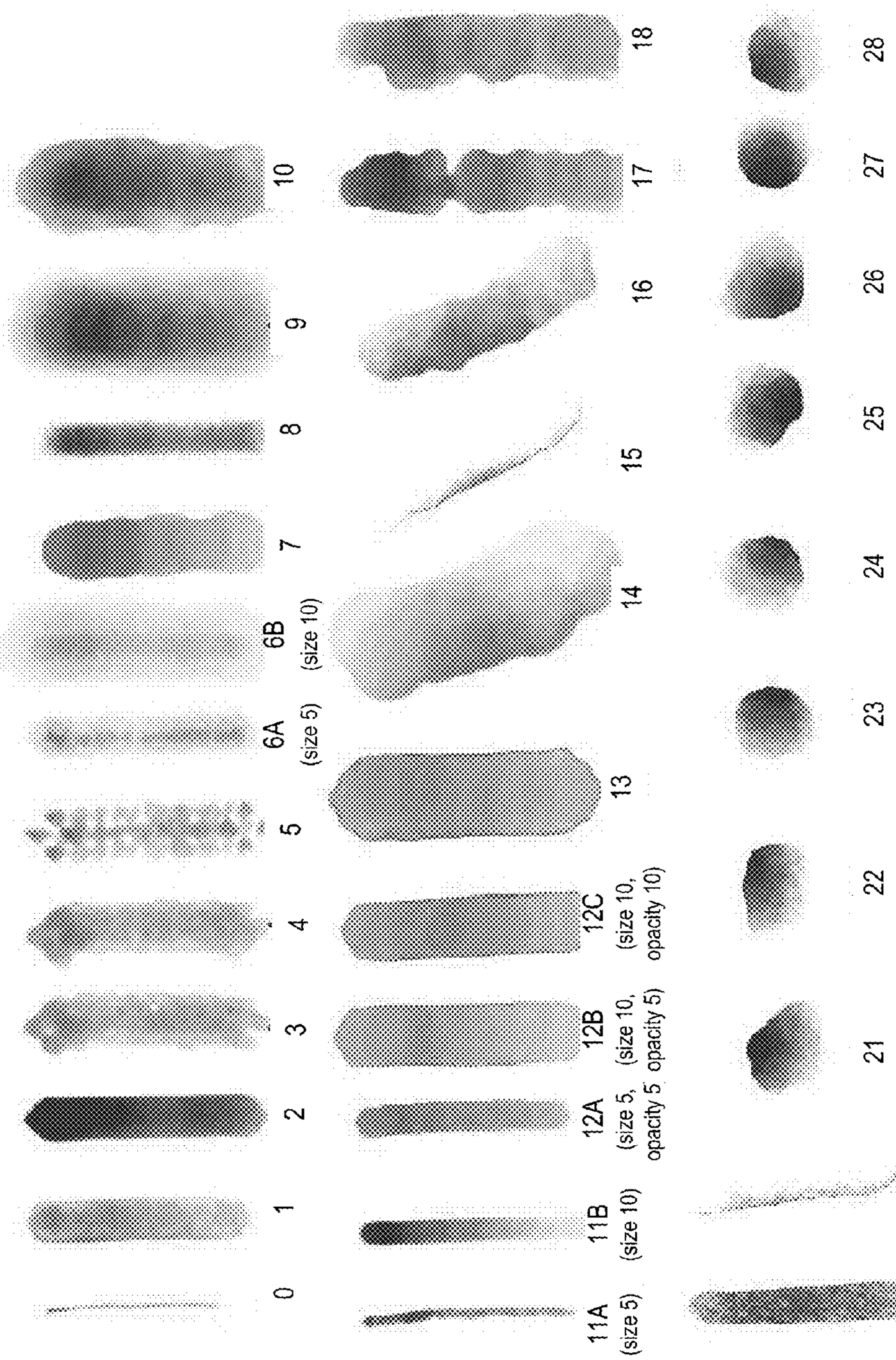

0
1
2
3
4
5
6A (size 5)
6B (size 10)
7
8
9
10
11A (size 5)
11B (size 10)
12A (size 5, opacity 5)
12B (size 10, opacity 5)
12C (size 10, opacity 10)
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28

FIG. 13

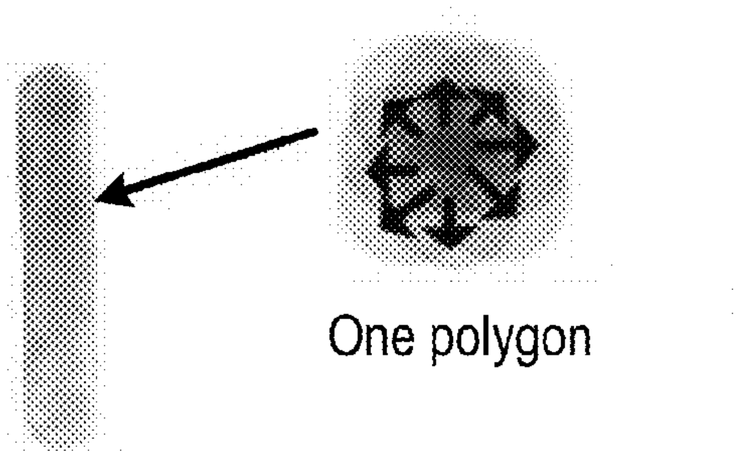


FIG. 14

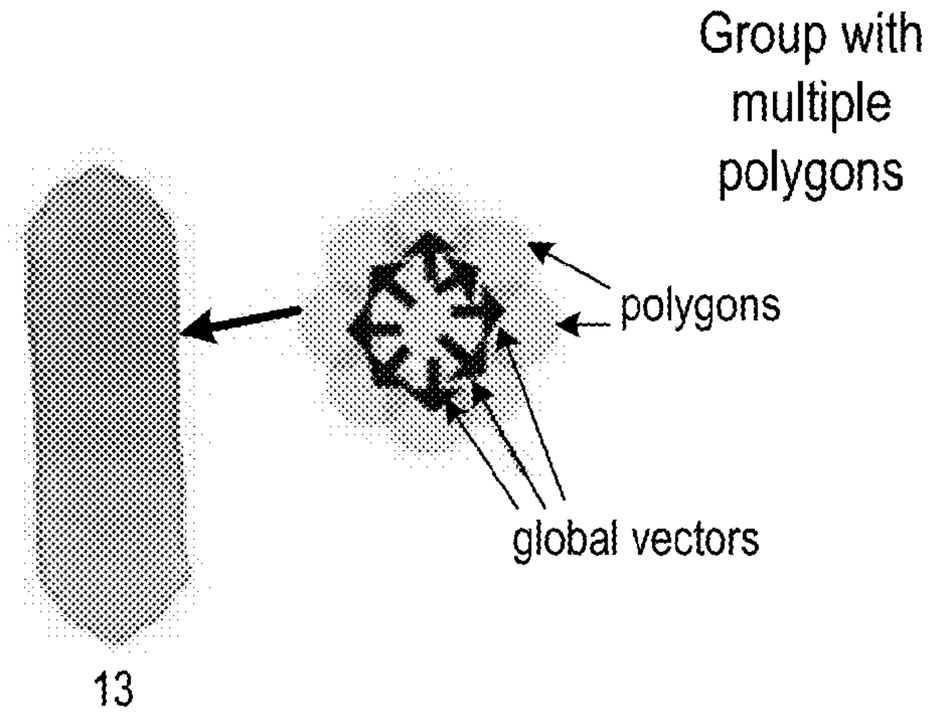


FIG. 15

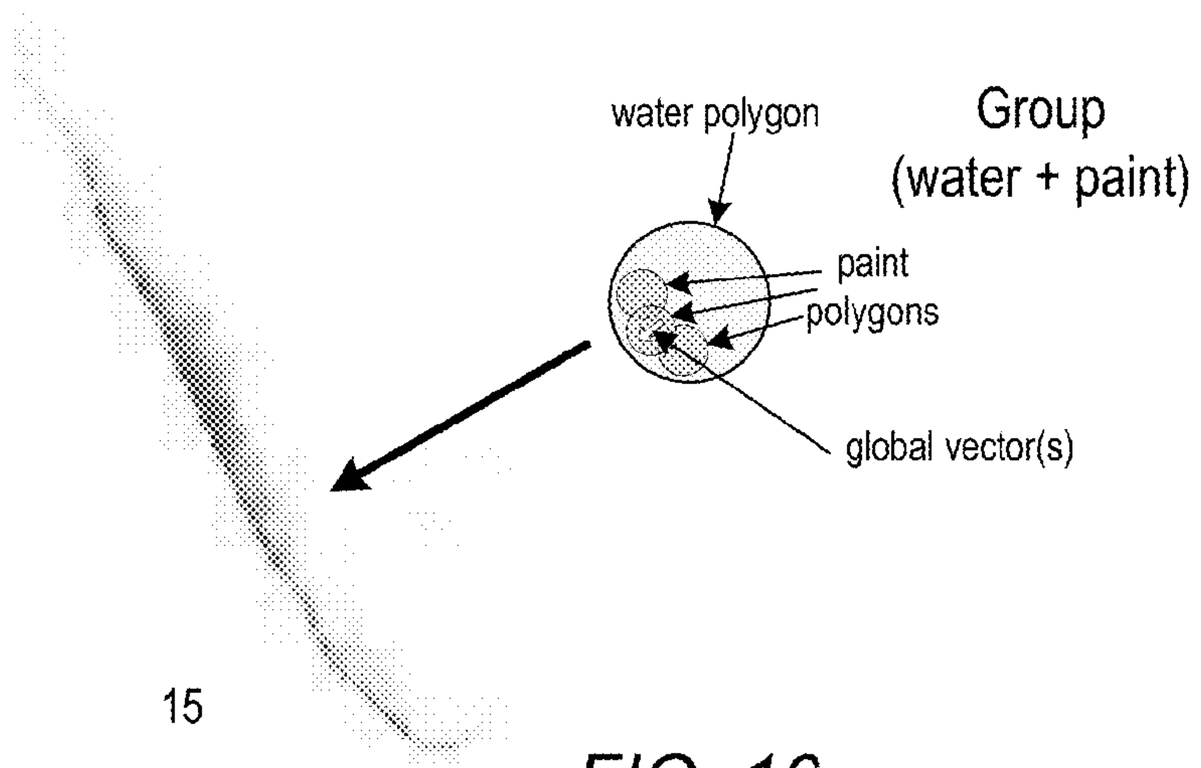


FIG. 16

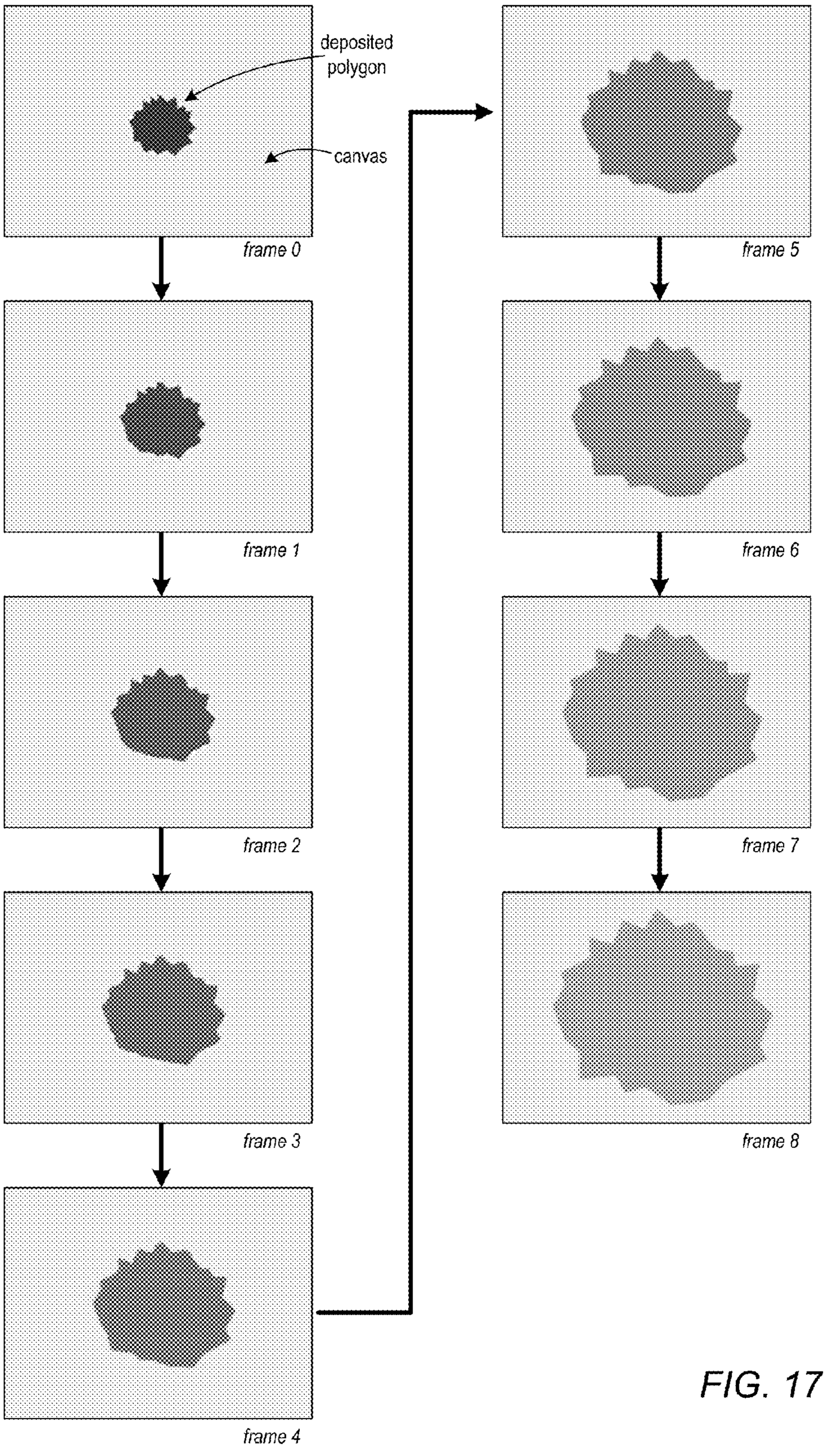

deposited polygon
canvas
frame 0
frame 1
frame 2
frame 3
frame 4
frame 5
frame 6
frame 7
frame 8

FIG. 17

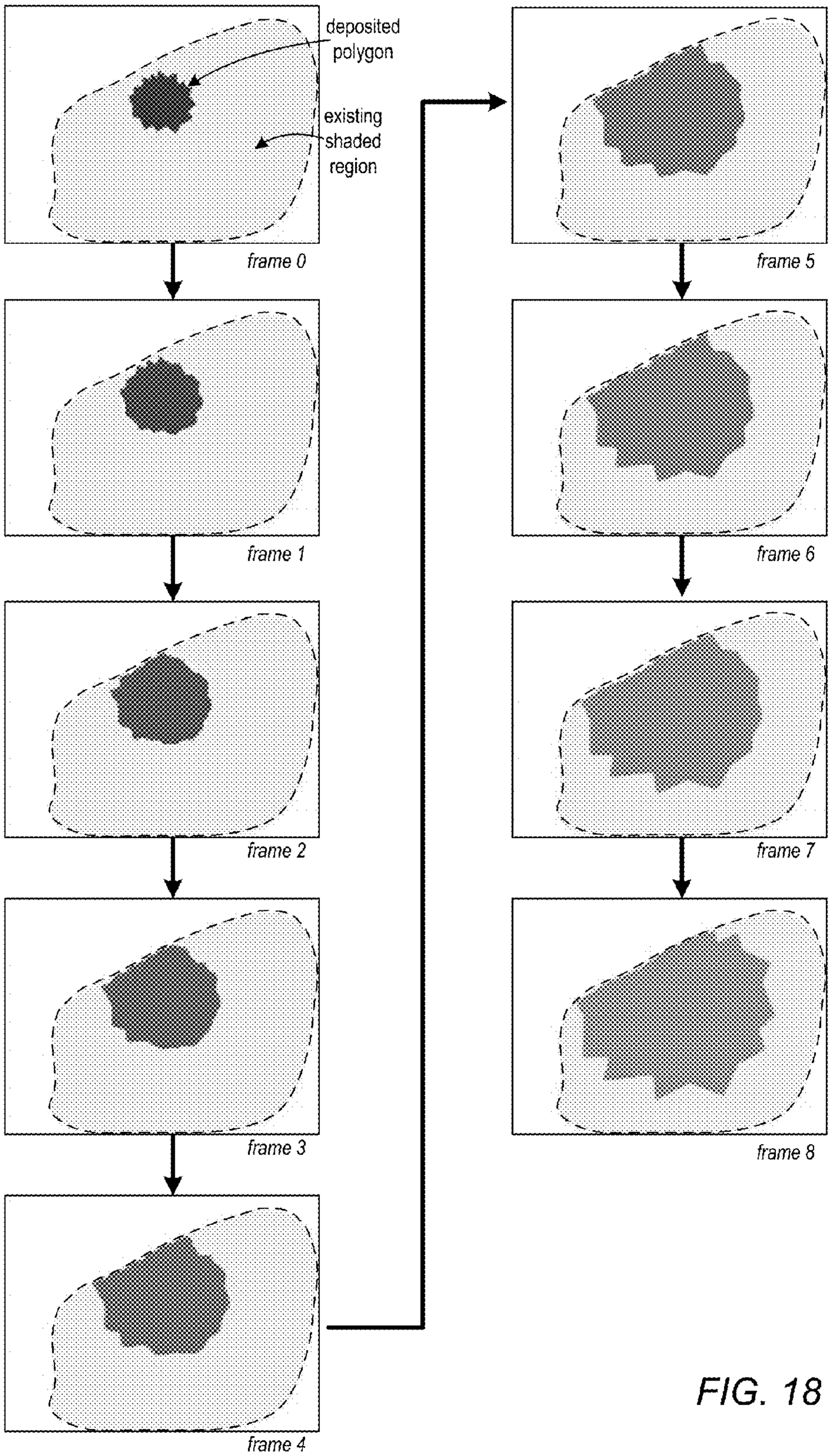

deposited polygon
existing shaded region
frame 0
frame 1
frame 2
frame 3
frame 4
frame 5
frame 6
frame 7
frame 8

FIG. 18

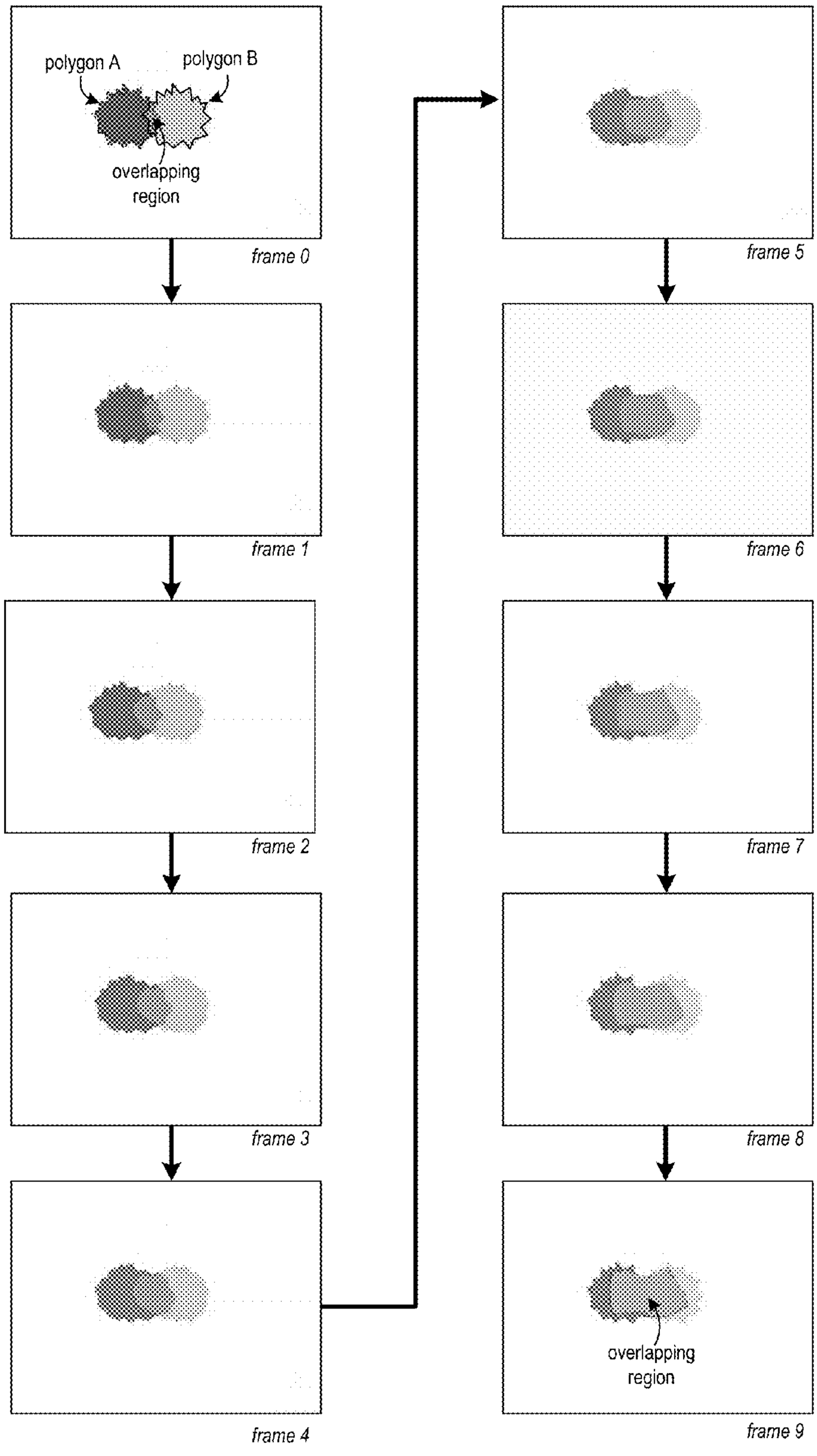

polygon A
polygon B
overlapping region
frame 0
frame 1
frame 2
frame 3
frame 4
frame 5
frame 6
frame 7
frame 8
frame 9
overlapping region

FIG. 19

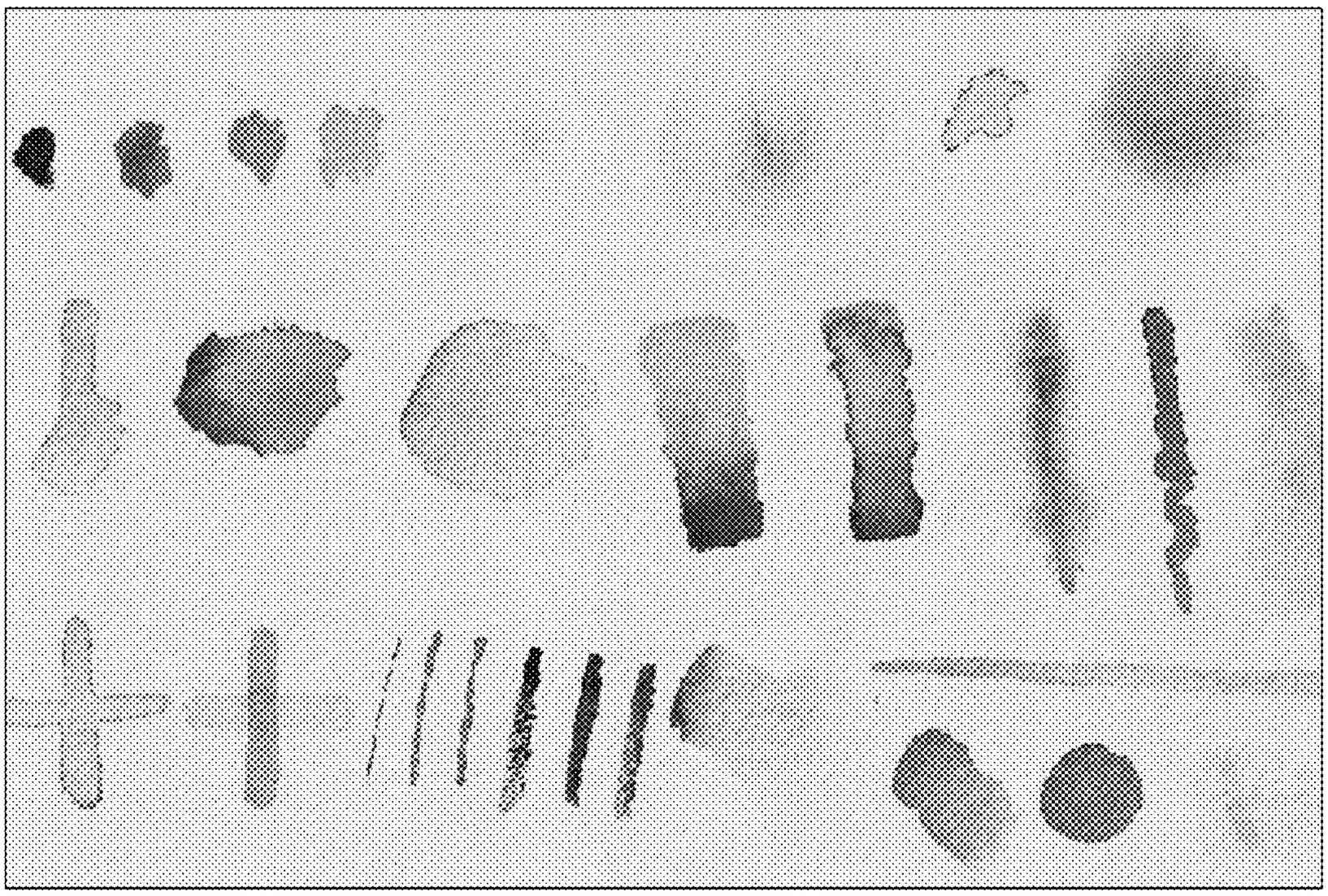


FIG. 20A

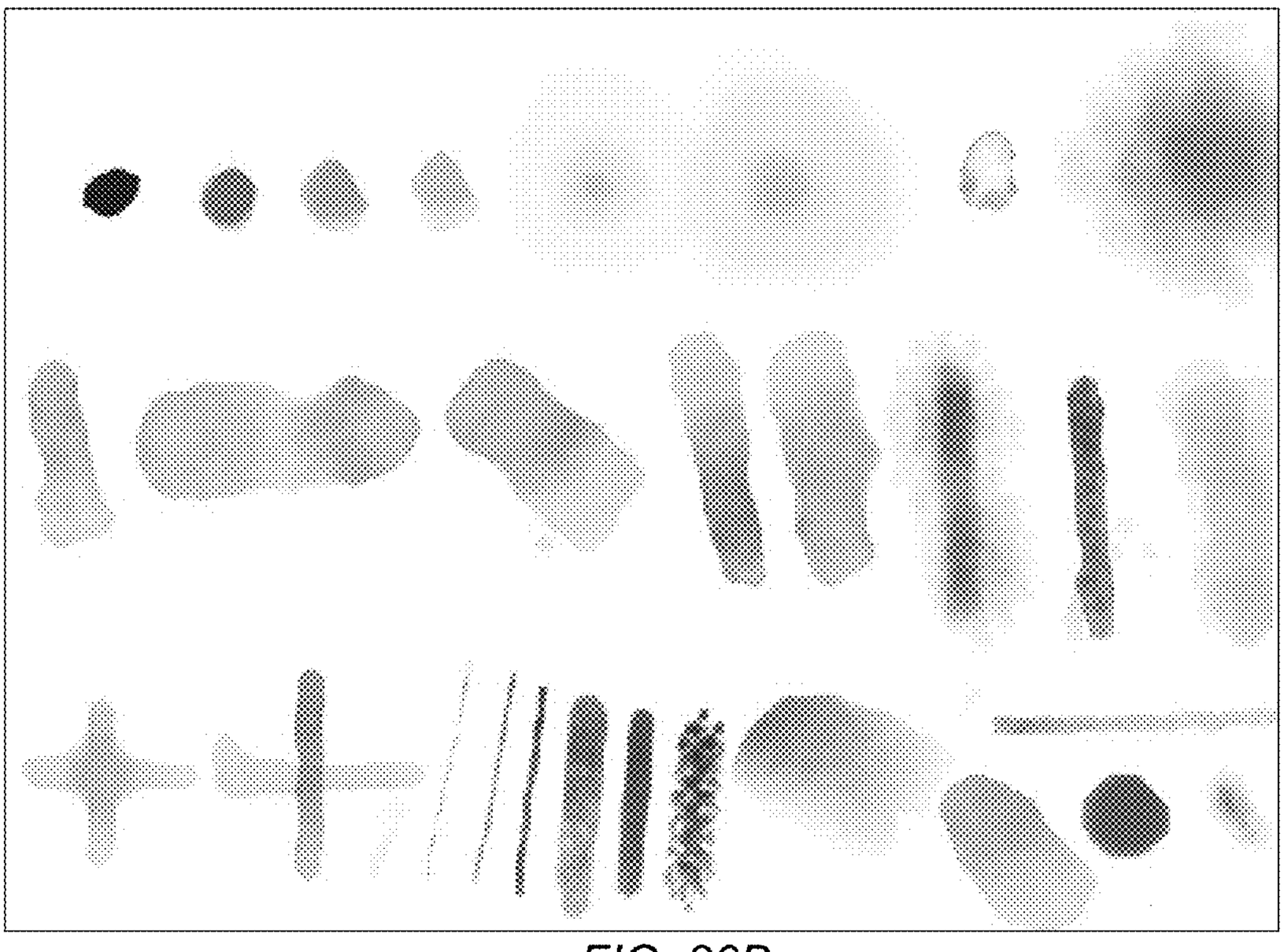


FIG. 20B

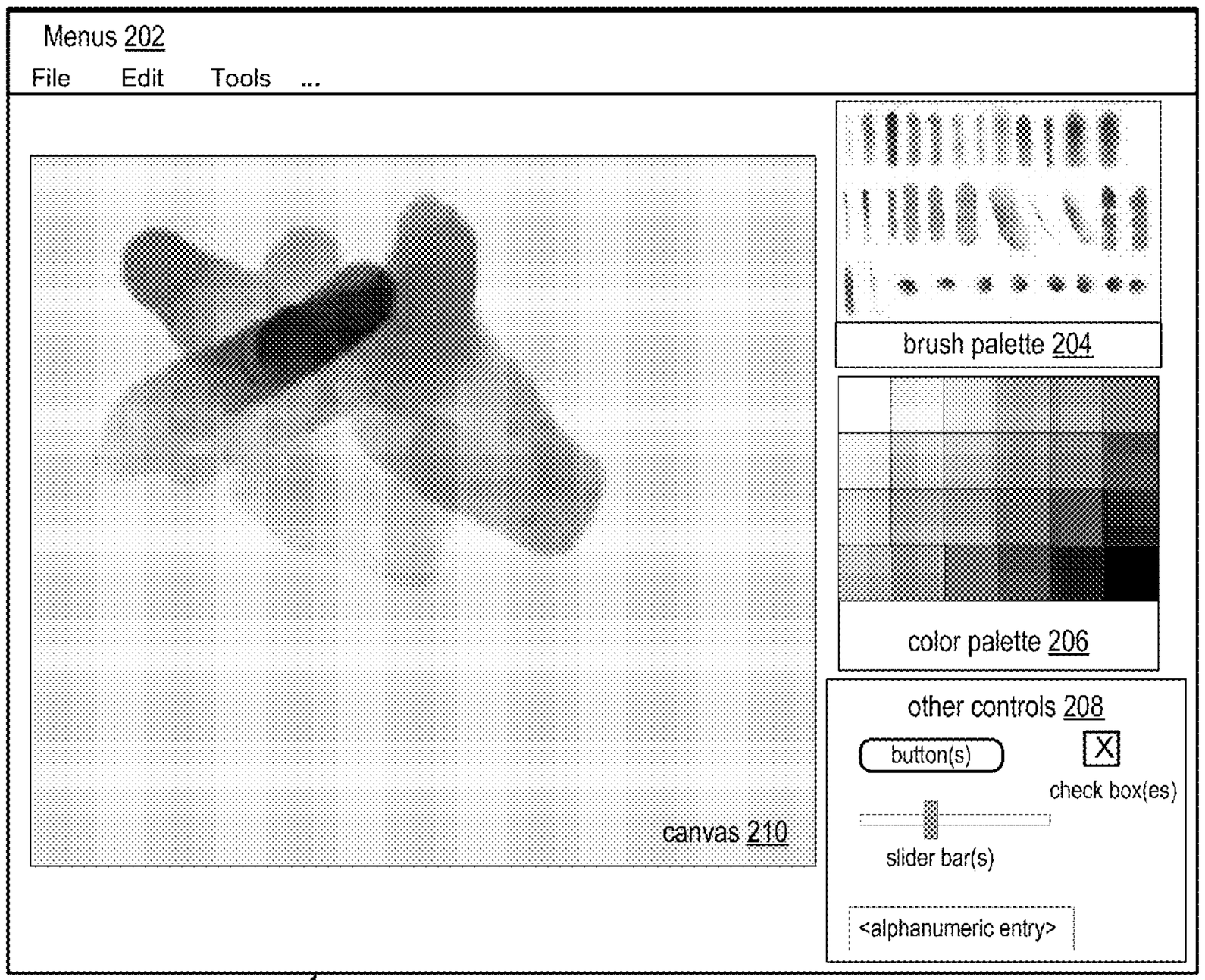


FIG. 21

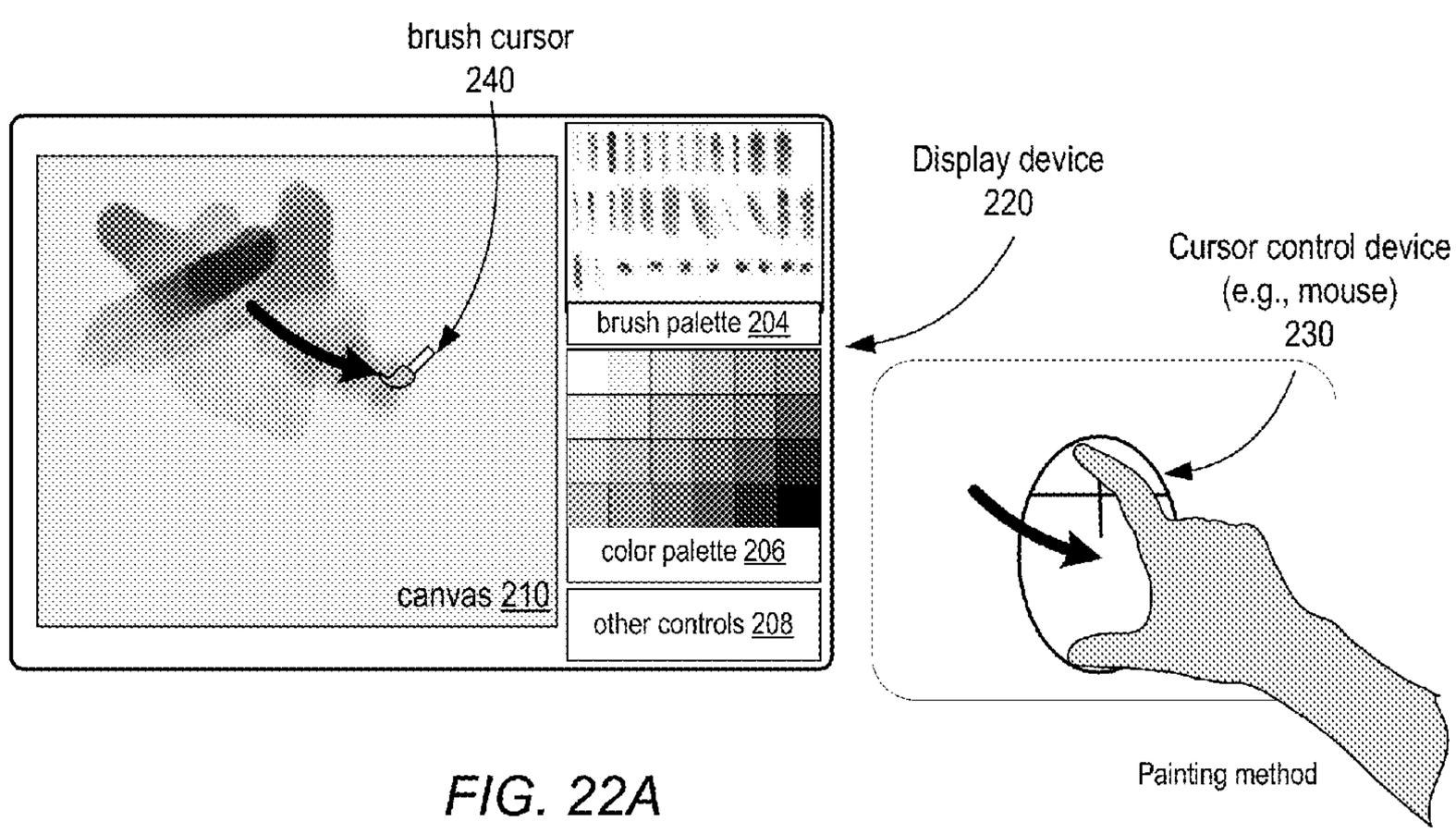


FIG. 22A

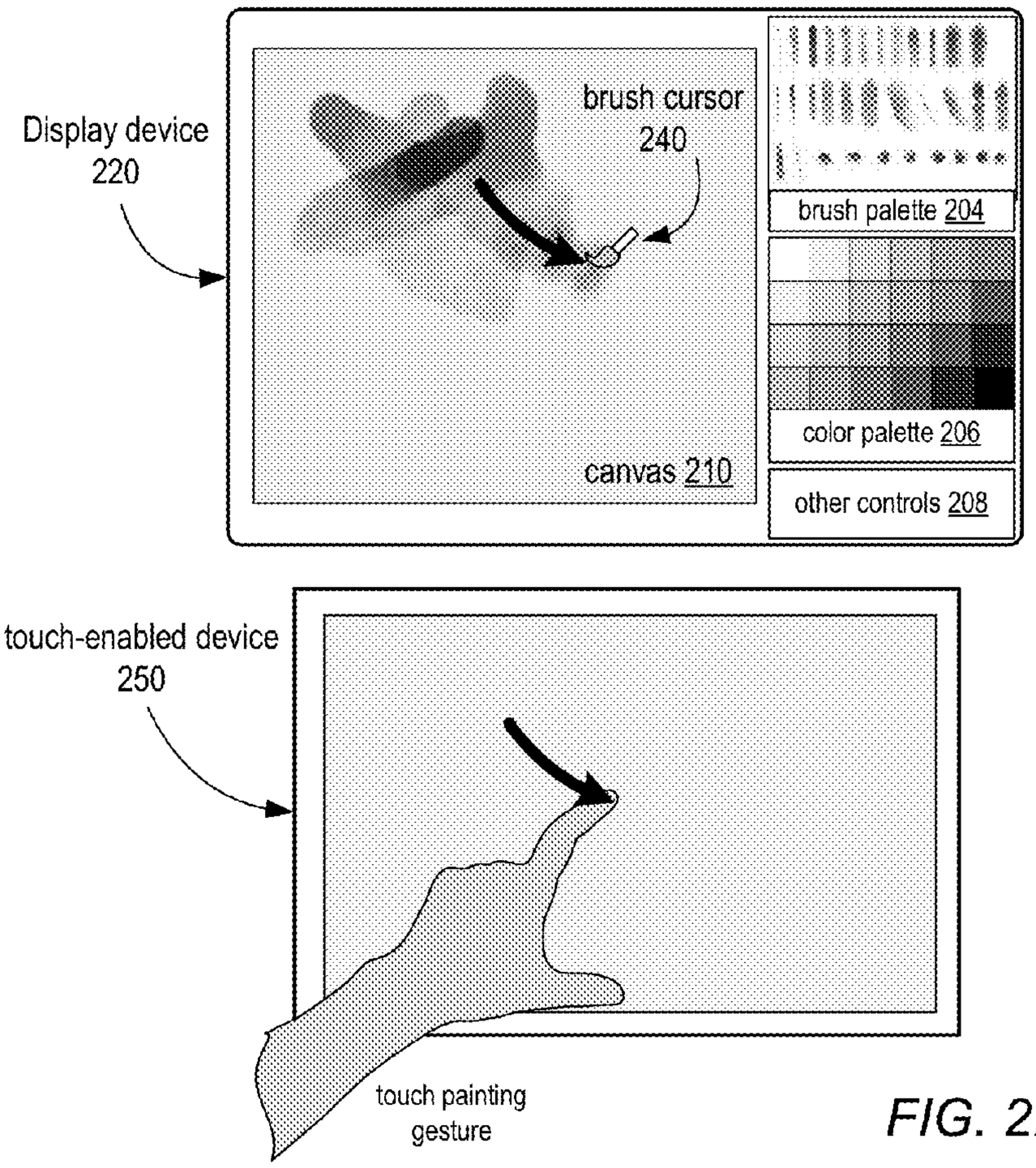


FIG. 22B

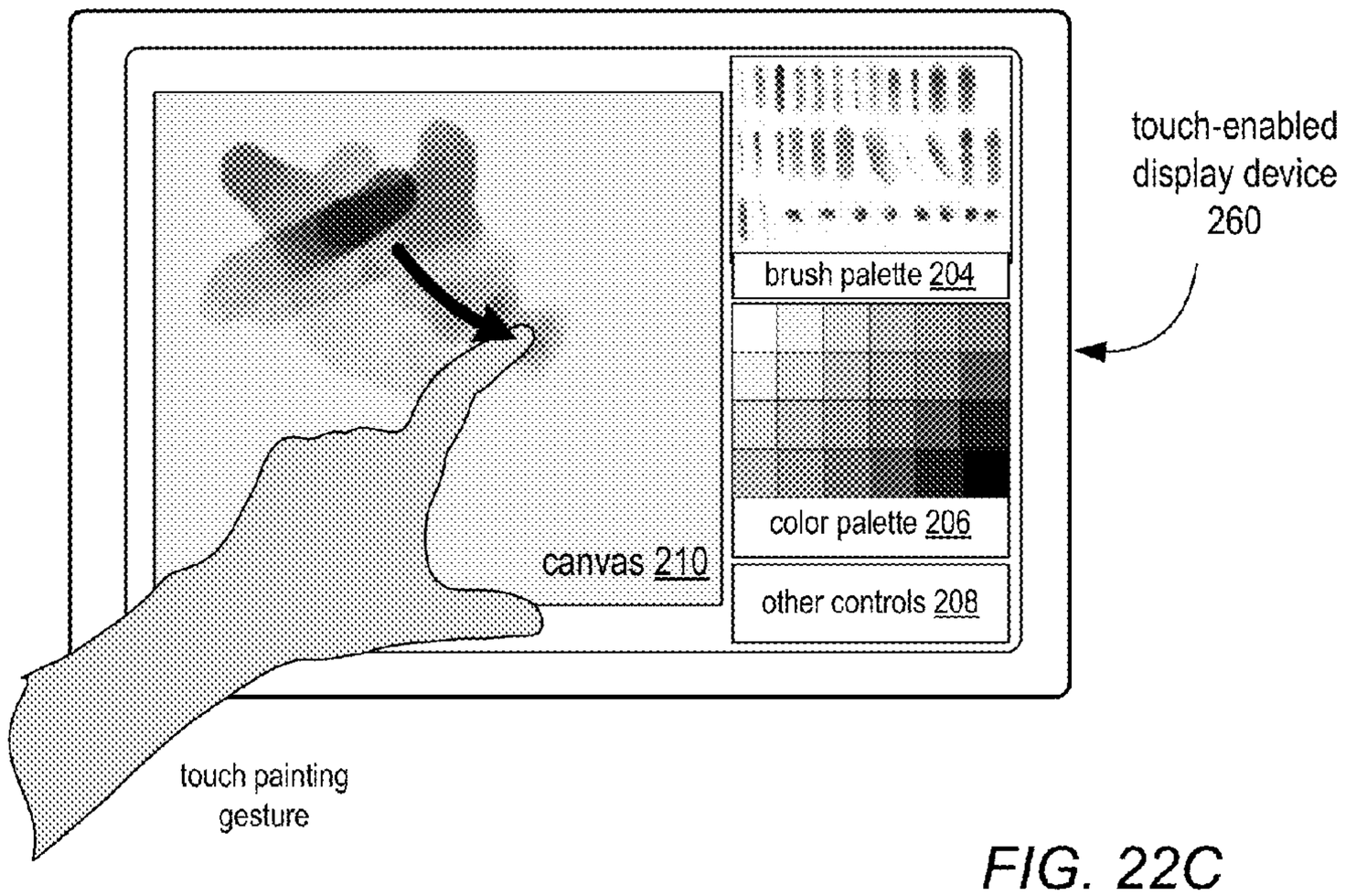


FIG. 22C

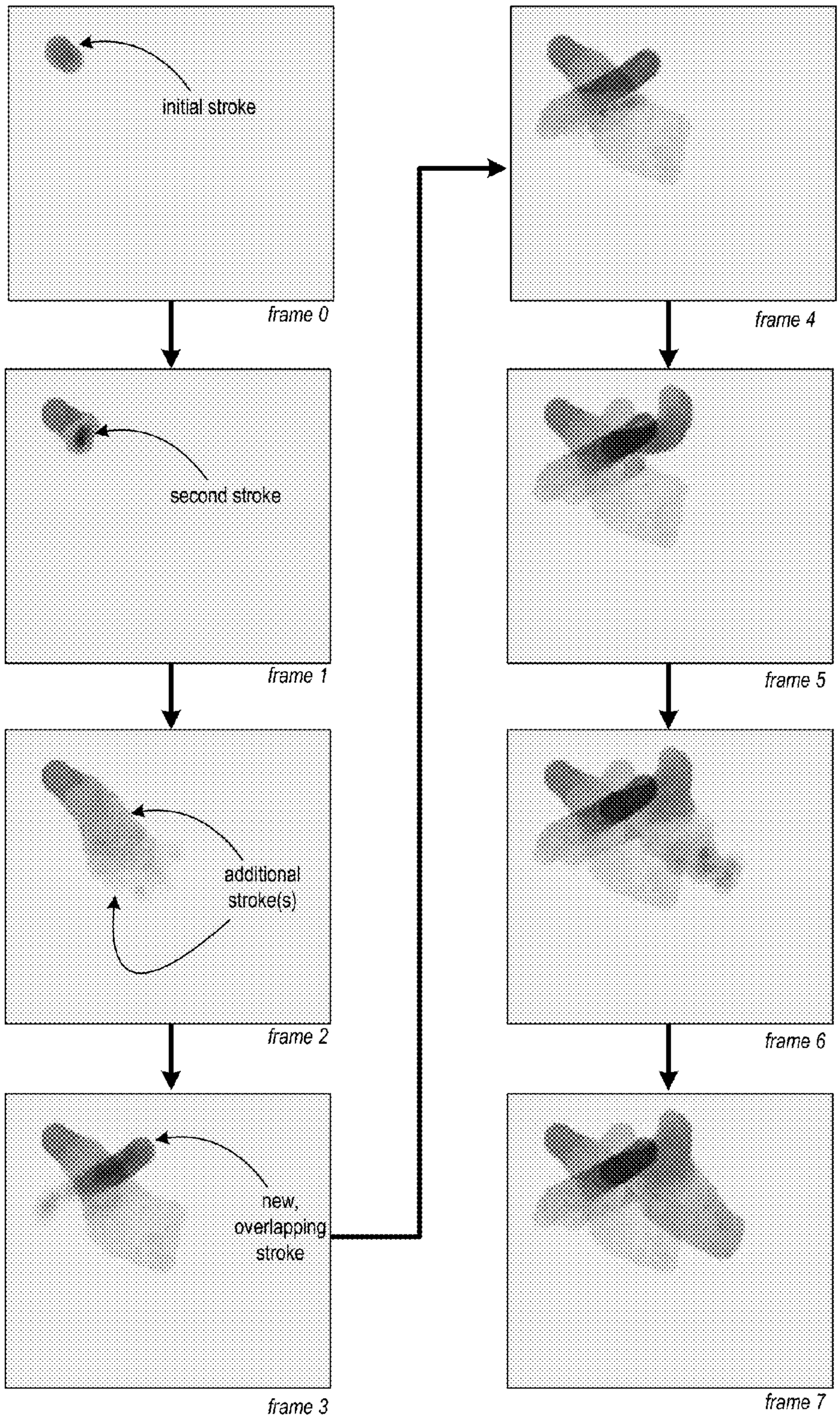

initial stroke
frame 0
second stroke
frame 1
additional stroke(s)
frame 2
new, overlapping stroke
frame 3
frame 4
frame 5
frame 6
frame 7

FIG. 23

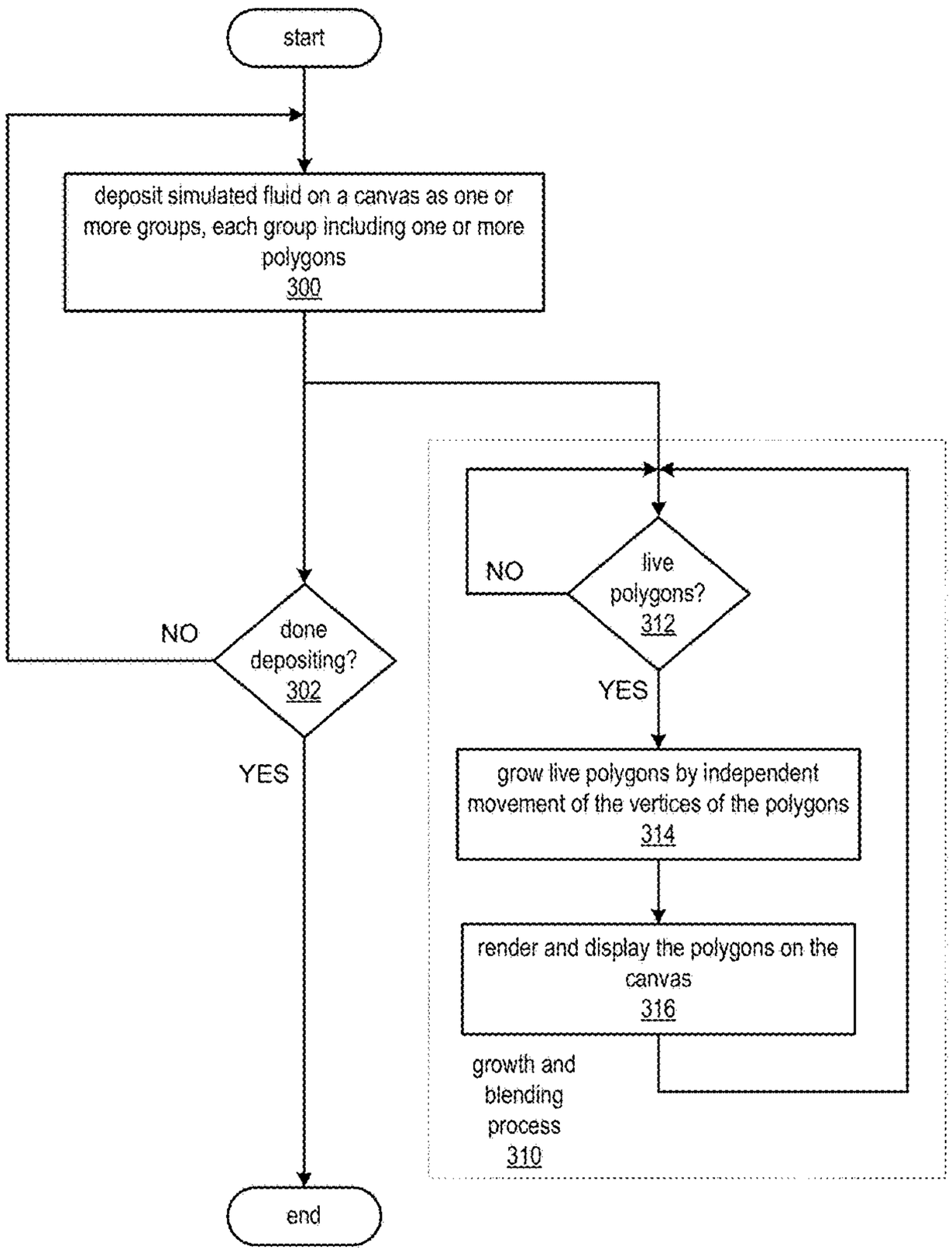

start
deposit simulated fluid on a canvas as one or more groups, each group including one or more polygons
300
done depositing?
302
NO
YES
live polygons?
312
NO
YES
grow live polygons by independent movement of the vertices of the polygons
314
render and display the polygons on the canvas
316
growth and blending process
310
end

FIG. 24

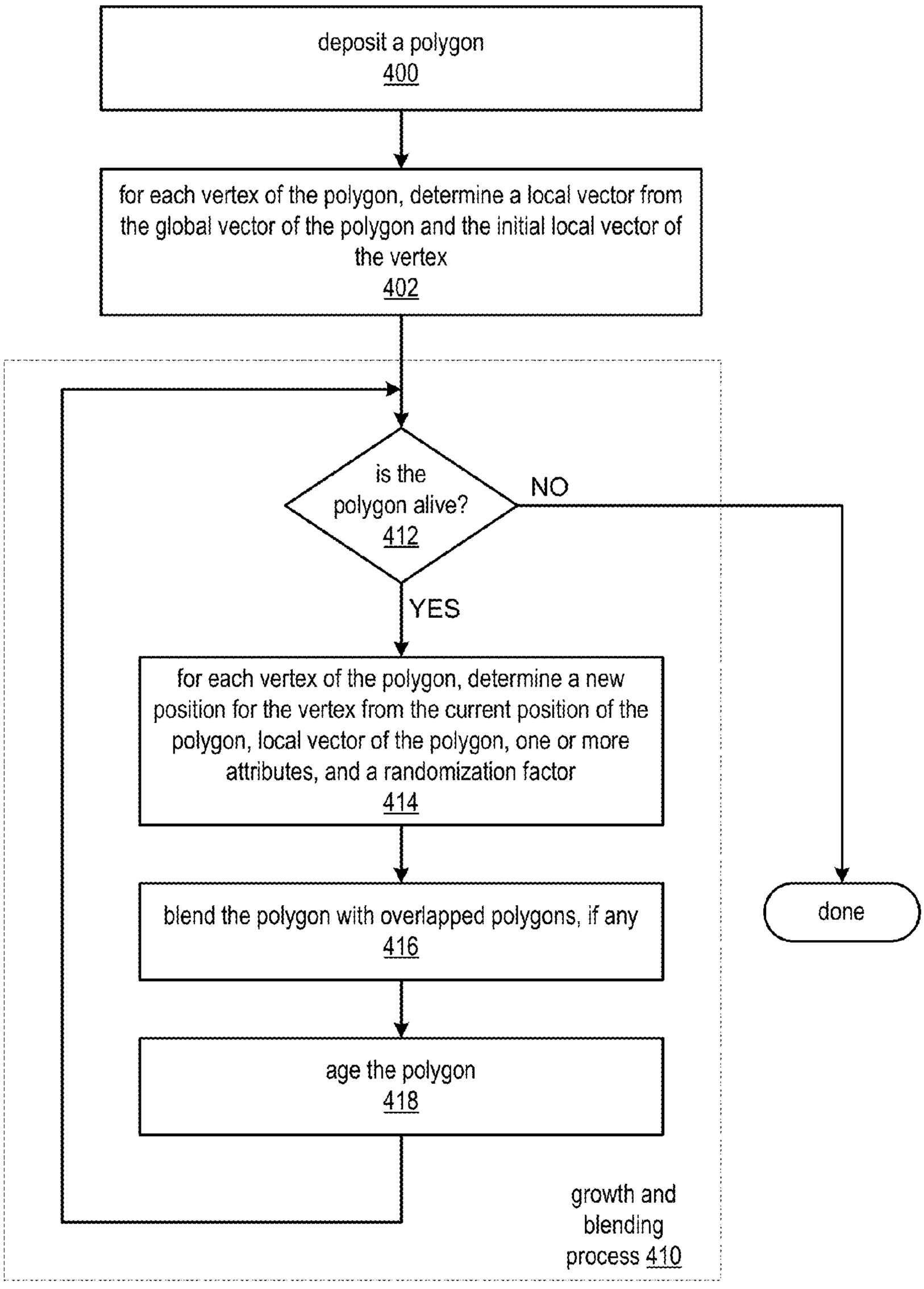

deposit a polygon
400
for each vertex of the polygon, determine a local vector from the global vector of the polygon and the initial local vector of the vertex
402
is the polygon alive?
412
NO
YES
for each vertex of the polygon, determine a new position for the vertex from the current position of the polygon, local vector of the polygon, one or more attributes, and a randomization factor
414
blend the polygon with overlapped polygons, if any
416
age the polygon
418
done
growth and blending process 410

FIG. 25

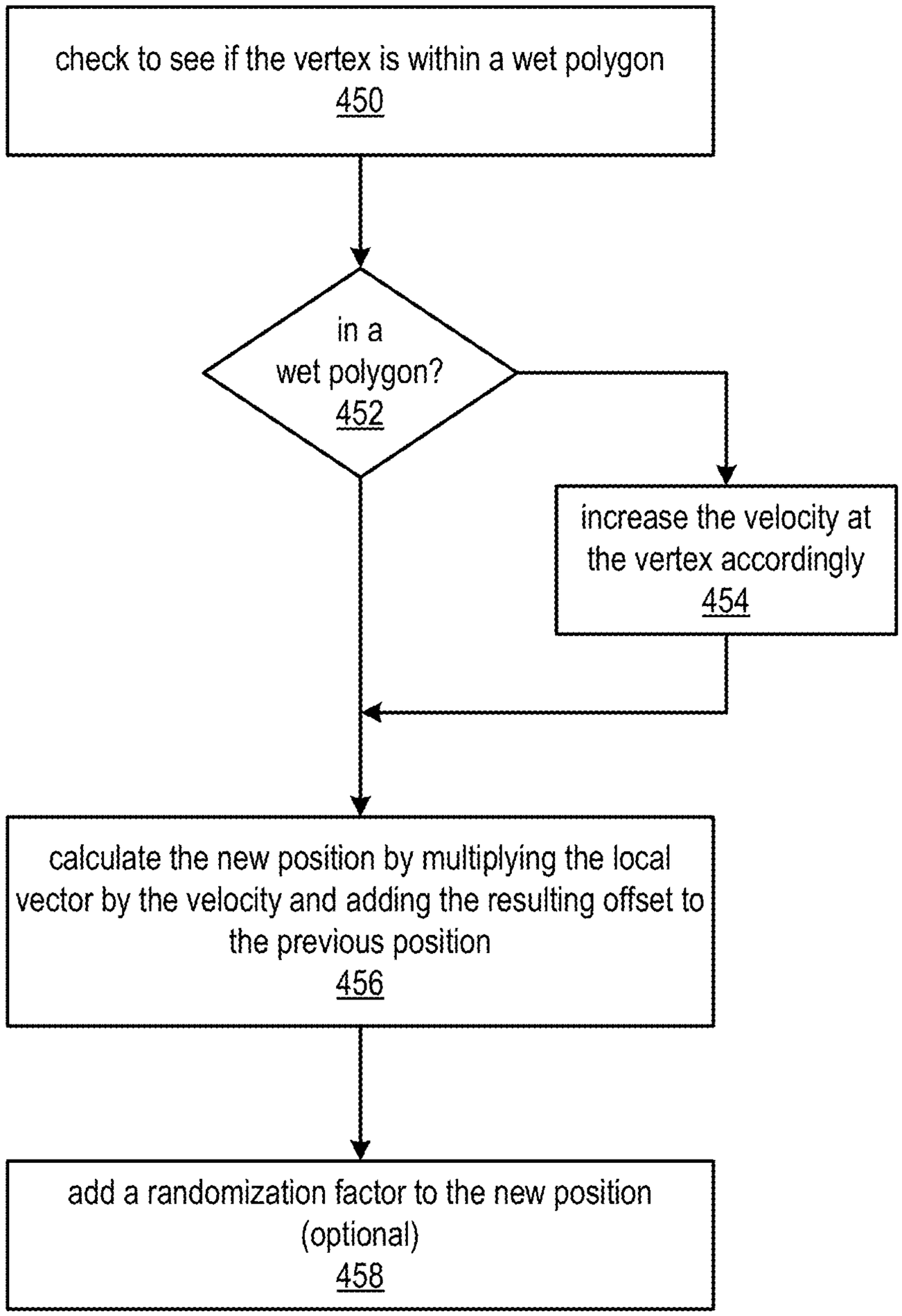

check to see if the vertex is within a wet polygon
450
in a wet polygon?
452
increase the velocity at the vertex accordingly
454
calculate the new position by multiplying the local vector by the velocity and adding the resulting offset to the previous position
456
add a randomization factor to the new position (optional)
458

FIG. 26

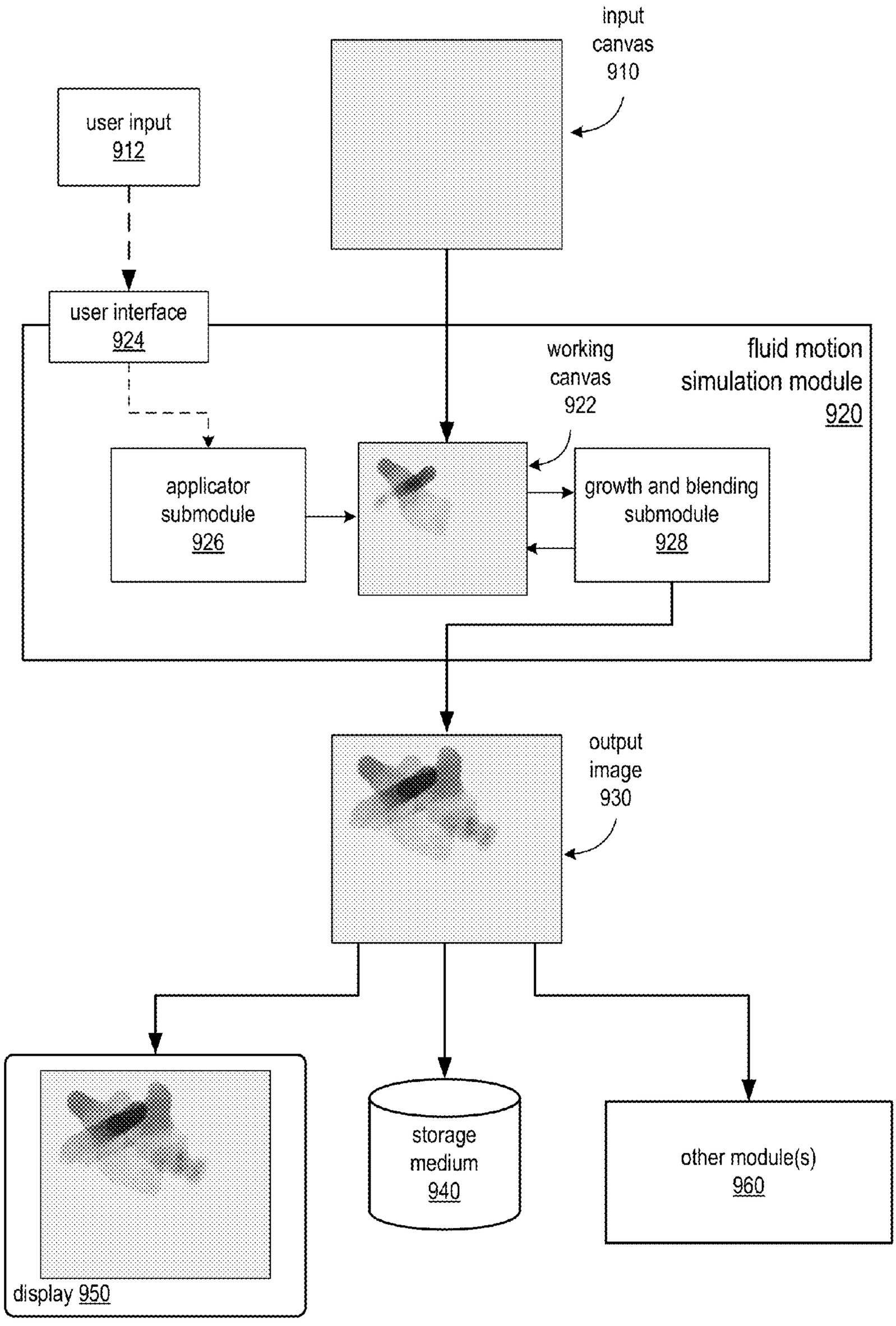

input canvas 910
user input 912
user interface 924
working canvas 922
fluid motion simulation module 920
applicator submodule 926
growth and blending submodule 928
output image 930
display 950
storage medium 940
other module(s) 960

FIG. 27

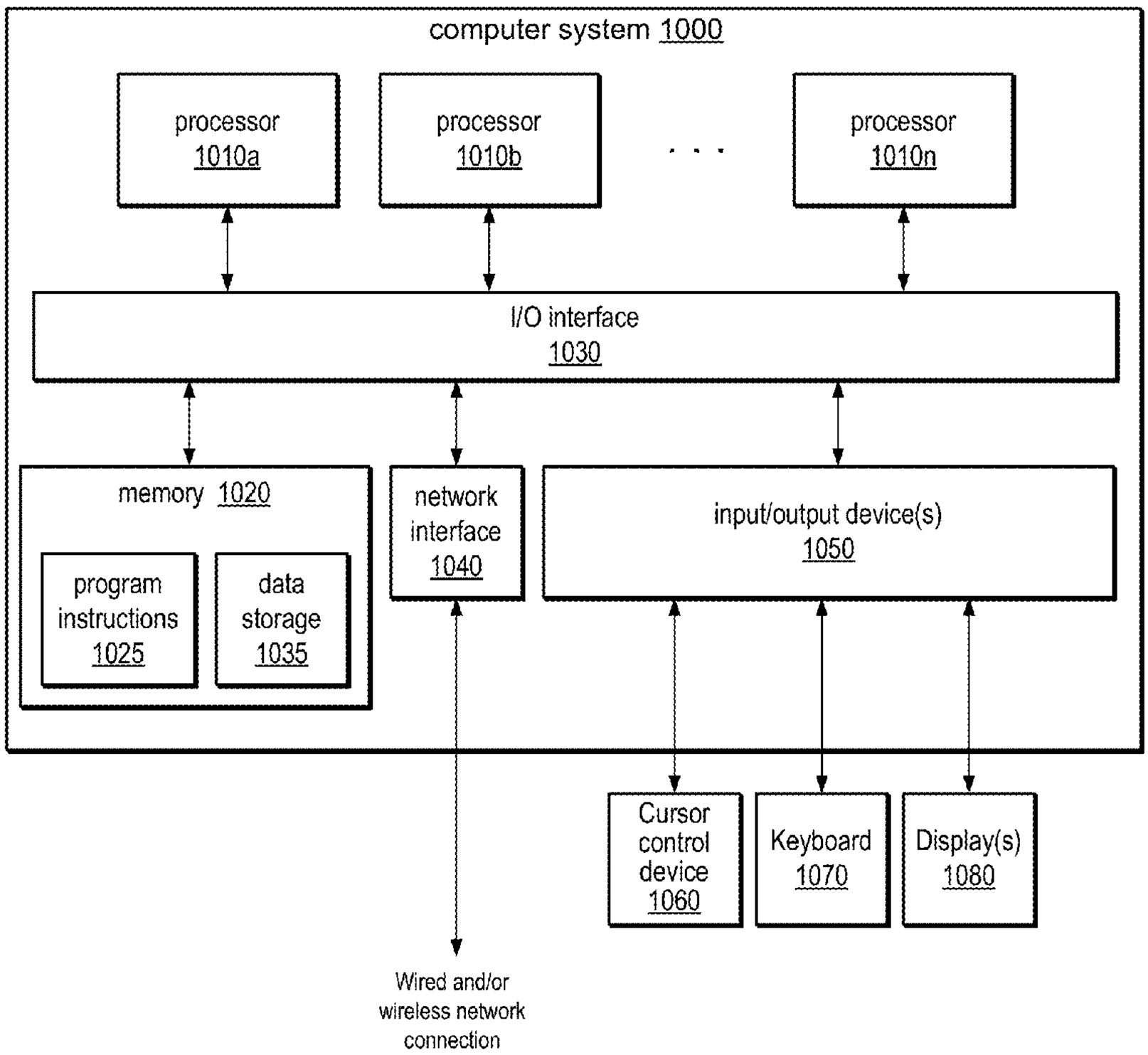

computer system 1000
processor 1010a
processor 1010b
processor 1010n
I/O interface 1030
memory 1020
program instructions 1025
data storage 1035
network interface 1040
input/output device(s) 1050
Cursor control device 1060
Keyboard 1070
Display(s) 1080
Wired and/or wireless network connection

FIG. 28

METHODS AND APPARATUS FOR SIMULATION OF FLUID MOTION USING PROCEDURAL SHAPE GROWTH

BACKGROUND

Description of the Related Art

Conventionally, the simulation of fluid motion in computer graphics has been handled using raster-based techniques. That is, fluids and gasses have been simulated on a grid of cells or pixels, and fluid motion is computed between pixels on the grid. These conventional techniques involve processing each pixel on the grid, where neighboring pixels are examined and one or more fluid dynamics equations are computed at the pixel. In high-resolution applications in particular, such as high-resolution painting applications, a large number of pixels may have to be processed at each of multiple iterations. Even if the fluid motion effect being simulated is relatively simple and thus the computation at each pixel is relatively straightforward, this still involves considerable computation, and thus time, due to the large number of pixels being individually processed. If more complicated fluid motion effects are being simulated, the computations at each pixel become more complicated, and thus the processing may take much longer.

SUMMARY

Various embodiments of methods and apparatus for simulating fluid motion using procedural shape growth are described. Embodiments of a vector-based, fluid motion simulation technique are described in which fluid location may be defined by groups of one or more polygons deposited on a digital canvas. Two or more polygons may overlap. The polygons may generally be, but are not necessarily, semitransparent. A set of polygons may be, but are not necessarily, of uniform color. To simulate fluid motion, instead of computing the fluid flow between cells or pixels of a grid as in conventional raster-based techniques, embodiments instead compute the movement of vertices that specify the edges of each deposited polygon. By moving the vertices, a polygon may grow, and fluid motion effects may be simulated, including but not limited to directional flow and blending effects. Embodiments of the vector-based, fluid motion simulation technique may require less processing power than conventional, raster-based techniques that compute flow and other effects at each cell or pixel, and thus may render frames faster than conventional, raster-based fluid motion simulators. In addition, raster-based techniques tend to exhibit pixilation when zoomed in. Since the fluid motion simulation technique is vector-based, objects drawn with the technique may exhibit smooth edges when zoomed into.

Embodiments of the vector-based, fluid motion simulation technique may, for example, be used in simulated painting applications, including but not limited to simulated watercolor painting applications. In embodiments of a painting method that employs the vector-based, fluid motion simulation technique, simulated fluid, for example simulated watercolor paint and/or water, may be deposited on a canvas as one or more groups, each group including one or more polygons. The polygons may be, but are not necessarily, semitransparent. For example, paint may be deposited to a digital canvas via strokes applied with a brush. In at least some embodiments, the deposited polygons may be iteratively grown and blended by a method implemented in a background process. Alternatively, the deposited polygons may be iteratively grown and blended by a method implemented in a programmatic loop, and thus not in a background process. At each iteration, the method may check to see if there are any live polygons according to the lifespan attribute of the polygons. If there are live polygons, then the method may grow the live polygons by independent movement of the vertices of the polygons. In at least some embodiments, a randomization technique may be applied to the movement at each vertex of each polygon. The method may then render and display the polygons on the canvas. Rendering and displaying the polygons may include blending overlapping semitransparent polygons with overlapped polygons or other overlapped objects, such as the canvas itself. Additional simulated fluid may be deposited to the canvas. For example, a user may choose to add additional paint to the canvas via the user interface. The additional deposited polygons may be similarly grown and blended by the background process or programmatic loop. The background process or programmatic loop may continue to iteratively process the deposited polygons until there are no more live polygons, that is until the paint is "dry."

In at least some embodiments of a method for processing a deposited polygon, a polygon may be deposited to a canvas. The deposited polygon may, but does not necessarily, overlap another polygon. At an initialization step, for each vertex of the polygon, a local vector may be determined from a global vector of the polygon and the initial local vector of the vertex. In at least some embodiments, the global vector of the polygon and the initial local vector of the vertex are summed to generate the starting local vector for the vertex. After initialization, the deposited polygon may be iteratively grown and blended by a method implemented in a background process or programmatic loop. At each iteration, the method may check to see if the polygon is alive according to a lifespan attribute of the polygon. If the polygon is alive, then the method may grow the polygon by independent movement of the vertices of the polygon. For each vertex of the polygon, a new position for the vertex may be determined from the current position of the polygon, the local vector of the polygon, one or more attributes, and a randomization factor. In at least some embodiments, at each vertex, the following equation may be used to compute the new position to which the vertex is to be moved:

$$P_{new}=P_{old}+V*S+R$$

where P_{old} designates the current position of the vertex, P_{new} designates the new position for the vertex, V designates the local vector for the vertex, S designates the speed or velocity at the vertex, and R represents randomized values that may be added to the (x,y) coordinates of the vertex; the randomized values may be positive or negative. In at least some embodiments, at each vertex of a polygon, before computing P_{new} for the vertex, a check may be performed to determine if the vertex is on a wet polygon that is overlapped by the vertex's polygon. If the vertex is on a wet polygon, then S (the speed, or velocity) at the vertex may be increased to account for the wetness of the overlapped polygon.

The polygon may be blended with overlapped polygons, if any. In at least some embodiments, to blend a polygon with an overlapped polygon, a blending technique may be applied in which the color of the polygon is multiplied by an alpha value and added to the color of the overlapped polygon, multiplied by (1−alpha). In at least some embodiments, alpha may be a floating-point number, for example in the range [0,1], and may be derived from, the current opacity level of the overlapping polygon.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1**A illustrates the border of an example polygon, according to at least some embodiments.

FIG. **1**B shows an example of shading that may be displayed on a canvas when the example polygon of FIG. **1**A is initially applied, according to at least some embodiments.

FIG. **1**C shows an example in which two of the example polygon from FIGS. **1**A and **1**B are applied so as to partially overlap, according to at least some embodiments.

FIGS. **2**A through **2**D show various examples of polygons that may be used, according to at least some embodiments.

FIG. **3** illustrates global and local attributes of an example polygon, according to at least some embodiments.

FIG. **4** illustrates the application of four instances of the example polygon from FIGS. **1**A and **1**B to a canvas, according to some embodiments.

FIG. **5** illustrates an example group that includes three partially overlapping, semitransparent polygons, according to at least some embodiments.

FIG. **6** illustrates an example group that includes five overlapping polygons, according to at least some embodiments.

FIG. **7** illustrates an example group that includes nine overlapping polygons, according to at least some embodiments.

FIG. **8** illustrates an example group that includes the example polygon of FIG. **2**A in the center, surrounded by four polygons similar to the example polygon of FIG. **2**C.

FIG. **9** illustrates an example group that includes a six-sided polygon in the center, surrounded by six polygons similar to the example polygon of FIG. **2**C.

FIG. **10** illustrates an example group that includes a relatively large polygon with three partially overlapping smaller polygons arranged along one side of the larger polygon.

FIG. **11** illustrates an example group in which a larger polygon completely overlaps or encloses a smaller polygon.

FIG. **12** illustrates the application of eight instances of the example group from FIG. **6** to a canvas, according to at least some embodiments.

FIG. **13** illustrates a set of example brush strokes made by example brushes that may be defined, for example for a watercolor simulation application, according to at least some embodiments.

FIG. **14** shows example brush **1** from FIG. **13** as including a group with one substantially circular polygon.

FIG. **15** shows example brush **13** from FIG. **13** as including a group with multiple substantially circular polygons.

FIG. **16** shows example brush **15** from FIG. **13** as including a group with multiple substantially circular polygons, where one of the polygons is water and the others are pigment.

FIG. **17** illustrates the growth and blending of a polygon deposited on a digital canvas, according to at least some embodiments.

FIG. **18** illustrates the growth and blending of a polygon deposited on an existing painted region, according to at least some embodiments.

FIG. **19** illustrates the deposition, growth and blending of two partially overlapping polygons, according to at least some embodiments.

FIG. **20**A is an image that shows several example real strokes and effects achieved via natural watercolor painting.

FIG. **20**B illustrates each of the example real watercolor strokes in FIG. **20**A simulated with an embodiment of the vector-based, fluid motion simulation technique.

FIG. **21** illustrates a user interface that may be used with at least some embodiments of the vector-based, fluid motion simulation technique, for example in a watercolor simulation application.

FIGS. **22**A through **22**C illustrate several methods that may be used to interact with a user interface to the vector-based, fluid motion simulation technique to deposit paint on a canvas and to perform other actions, according to at least some embodiments.

FIG. **23** illustrates an example of watercolor painting using the vector-based, fluid motion simulation technique, according to at least some embodiments.

FIG. **24** is a high-level flowchart of a painting method that employs the vector-based, fluid motion simulation technique, according to at least some embodiments.

FIG. **25** is a flowchart of a method for processing a deposited polygon, according to at least some embodiments.

FIG. **26** is a flowchart of a method for moving a vertex of a polygon, according to at least some embodiments.

FIG. **27** illustrates a module that may implement a vector-based fluid motion simulation technique, according to at least some embodiments.

FIG. **28** illustrates an example computer system that may be used in embodiments.

While the invention is described herein by way of example for several embodiments and illustrative drawings, those skilled in the art will recognize that the invention is not limited to the embodiments or drawings described. It should be understood, that the drawings and detailed description thereto are not intended to limit the invention to the particular form disclosed, but on the contrary, the intention is to cover all modifications, equivalents and alternatives falling within the spirit and scope of the present invention. The headings used herein are for organizational purposes only and are not meant to be used to limit the scope of the description. As used throughout this application, the word "may" is used in a permissive sense (i.e., meaning having the potential to), rather than the mandatory sense (i.e., meaning must). Similarly, the words "include", "including", and "includes" mean including, but not limited to.

DETAILED DESCRIPTION OF EMBODIMENTS

In the following detailed description, numerous specific details are set forth to provide a thorough understanding of claimed subject matter. However, it will be understood by those skilled in the art that claimed subject matter may be practiced without these specific details. In other instances, methods, apparatuses or systems that would be known by one of ordinary skill have not been described in detail so as not to obscure claimed subject matter.

Some portions of the detailed description which follow are presented in terms of algorithms or symbolic representations of operations on binary digital signals stored within a memory of a specific apparatus or special purpose computing device or platform. In the context of this particular specification, the term specific apparatus or the like includes a general purpose computer once it is programmed to perform particular functions pursuant to instructions from program software. Algorithmic descriptions or symbolic representations are examples of techniques used by those of ordinary skill in the signal processing or related arts to convey the substance of their work to others skilled in the art. An algorithm is here, and is generally, considered to be a self-consistent sequence of operations or similar signal processing leading to a desired result. In this context, operations or processing involve physical manipulation of physical quantities. Typically, although

not necessarily, such quantities may take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared or otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to such signals as bits, data, values, elements, symbols, characters, terms, numbers, numerals or the like. It should be understood, however, that all of these or similar terms are to be associated with appropriate physical quantities and are merely convenient labels. Unless specifically stated otherwise, as apparent from the following discussion, it is appreciated that throughout this specification discussions utilizing terms such as "processing," "computing," "calculating," "determining" or the like refer to actions or processes of a specific apparatus, such as a special purpose computer or a similar special purpose electronic computing device. In the context of this specification, therefore, a special purpose computer or a similar special purpose electronic computing device is capable of manipulating or transforming signals, typically represented as physical electronic or magnetic quantities within memories, registers, or other information storage devices, transmission devices, or display devices of the special purpose computer or similar special purpose electronic computing device.

Various embodiments of methods and apparatus for simulating fluid motion using procedural shape growth are described. Embodiments of a vector-based, fluid motion simulation technique are described in which fluid location may be defined by groups of one or more polygons deposited on a digital canvas. Two or more polygons may overlap. The polygons may generally be, but are not necessarily, semi-transparent. A set of polygons may be, but are not necessarily, of uniform color. To simulate fluid motion, instead of computing the fluid flow between cells or pixels of a grid as in conventional raster-based techniques, embodiments instead compute the movement of vertices that specify the edges of each deposited polygon. By moving the vertices, a polygon may grow, and fluid motion effects may be simulated, including but not limited to directional flow and blending effects. Each polygon may have a certain lifespan and a certain time in which it is considered wet. The movement of polygon vertices may be controlled or influenced by various attributes, including but not limited to whether the vertex is over another polygon that is still wet and an initial vector that may depend on the location of a particular polygon within an area of the initial deposition.

Embodiments of the vector-based, fluid motion simulation technique may require less processing power than conventional, raster-based techniques that compute flow and other effects at each cell or pixel, and thus may render frames faster than conventional, raster-based fluid motion simulators. In addition, raster-based techniques tend to exhibit pixilation when zoomed in. Since the fluid motion simulation technique is vector-based, objects drawn with the technique may exhibit smooth edges when zoomed into. In at least some embodiments, locally smoothed Bezier curves may be used for the edges between the vertices of the polygons to help achieve a smooth effect.

An example application for embodiments of the vector-based, fluid motion simulation technique is in watercolor painting simulation. However, the technique may be used for simulating the deposition, fluid motion, and blending of other substances (e.g. fluids, gasses, vapors, etc.) such as oil or acrylic paints, flowing water, smoke, clouds, fire, lava, and in general any other substance or phenomenon that exhibits similar dynamics.

In at least some embodiments, a fluid may be represented as a set of overlapping, semitransparent polygons of uniform color. The polygons may be deposited singly or in groups. Together, the deposited polygons capture the representation of the fluid across space. In watercolor simulation, for example, as a brush is moved across a canvas, watercolor paint (pigment plus water) may be deposited in groups of five partially overlapping polygons (see, e.g., FIGS. **6** and **8**). Five polygons in a group is an example; more or fewer polygons may be deposited in a group (see, e.g., FIGS. **5**, **7**, **9**, **10**, and **11**). Two or more polygons in a group may overlap, and groups may be deposited so that polygons in adjacent groups at least partially overlap. Fluid motion may be simulated by moving the vertices of the polygons representing the deposited fluid. In at least some embodiments, the movement of the vertices may be randomized. However, the movement of a vertex may be influenced by one or more factors determined by attributes of the respective polygon and/or group. For example, in watercolor simulation, each group or each polygon may have a certain lifespan and a certain time in which it is considered wet. Thus, factors that may influence the movement of vertices may include whether a given vertex in one (overlapping) polygon is over another (overlapped) polygon that is still wet. Other factors that may influence the movement of a vertex may include an initial velocity and an initial, global direction vector. Attributes such as lifespan, initial opacity, initial velocity, and initial vector may be associated with each polygon; different polygons in a group may have different attributes. Alternatively, one or more attributes may be associated with a group and inherited by or applied to the polygons in the group, and thus may influence at least the initial motion of all vertices of all polygons in the group.

Embodiments of the vector-based, fluid motion simulation technique may be implemented on various types of devices, including, but not limited to, a personal computer system, desktop computer, laptop, notebook, or netbook computer, mainframe computer system, handheld computer, workstation, and network computer. FIG. **28** illustrates an example computer system that may implement various embodiments. By employing a vector-based technique that simulates fluid motion by computing the movement of vertices that specify the edges of each deposited polygon, embodiments may require less processing power than conventional, raster-based techniques that compute flow and other effects at each cell or pixel, and thus may render frames faster than conventional fluid motion simulators. Thus, embodiments may be implemented on devices with relatively low processing power, such as tablet devices and PDAs, to render results of fluid deposition, motion, and blending faster than conventional techniques, making such applications practical. In general, embodiments may be implemented on any type of computing or electronic device to simulate fluid motion.

Embodiments of the vector-based, fluid motion simulation technique may be implemented in any artistic, graphical design, or image processing application, or more generally in any application in which fluid motion may be simulated for artistic or other purposes. Embodiments of the vector-based, fluid motion simulation technique may, for example, be implemented as a stand-alone application, as a module of an application, as a plug-in for applications including but not limited to artistic, graphical design, and image processing applications, and/or as a library function or functions that may be called by other applications such as artistic, graphical design, and image processing applications. Specific examples of applications or technologies in which embodiments may be implemented include, but are not limited to, Adobe® Photoshop® technology, Adobe® Flash® technology, and Adobe® After Effects® technology. "Adobe", "Photoshop",

"Flash", and "After Effects" are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Embodiments of the vector-based, fluid motion simulation technique may be performed by a fluid motion simulation module implemented by program instructions stored in a computer-readable storage medium and executable by one or more processors (e.g., one or more CPUs and/or GPUs). In some embodiments, at least some components of a fluid motion simulation module may be implemented on or in one or more graphics processing units (GPUs). An example fluid motion simulation module that may implement embodiments of the vector-based, fluid motion simulation technique as described herein is illustrated in FIG. **27**. An example computer system on which embodiments of the vector-based, fluid motion simulation technique may be implemented is illustrated in FIG. **28**.

Polygons

Embodiments of the vector-based, fluid motion simulation technique may employ polygons with various attributes; the polygons may be applied to a digital canvas in various arrangements via various techniques to achieve a wide range of fluid motion effects. The polygons may generally be, but are not necessarily, semitransparent. The border(s) of a polygon, defined by a set of vertices, may be expanded from an initial position when applied to the digital canvas to simulate the spread and blending of a deposited fluid. A polygon may have one or more global attributes, such as initial direction (as a global vector), velocity, and lifespan, that may affect the growth of the border. A polygon may have other attributes such as color and a position. In at least some embodiments, each vertex on the border of a polygon may have one or more local attributes, such as local direction (as a local vector) and velocity, that locally affect the movement of the individual vertex. In at least some embodiments, a randomization technique may be applied to the movement at each vertex to simulate the non-uniform expansion along the edges of a deposited fluid, for example in a watercolor simulation.

FIG. **1**A illustrates the border of an example polygon, according to at least some embodiments. Example vertices that define the polygon are shown, as well as example edges that connect the vertices. A polygon with irregular, jagged edges as shown in FIG. **1**A may be defined as an initial shape. However, the polygon shown in FIG. **1**A may also be representative of a smooth-edged polygon after one or more iterations of growing the border. In at least some embodiments, the edges of a polygon may be Bezier curves to facilitate smoothness, for example when zooming in. FIG. **1**B shows an example of shading that may be displayed on a canvas for the example polygon of FIG. **1**A.

Polygons may be individually applied to the digital canvas, or may be applied in groups that include two or more polygons. In this document, a group may be defined as a collection of one or more polygons that may be applied as a unit to a canvas. Two or more polygons applied to a digital canvas, for example in a brush stroke, may overlap. Generally, to achieve various fluid motion effects, polygons are applied so that two or more polygons do overlap. Overlapping may occur when one polygon is first applied, and a second polygon is then applied that overlaps the first polygon. In addition, two or more polygons in a group of polygons applied to the canvas as a unit may overlap. Two or more polygons that do not overlap when initially applied to the canvas may also grow until the polygons do overlap. FIG. **1**C shows an example in which two of the example polygon from FIGS. **1**A and **1**B are applied so as to partially overlap. The borders are shown in this example for clarity. Note the overlapping region.

A second polygon (or overlapping polygon) that overlaps a first polygon (or overlapped polygon) may be affected by the first polygon. For example, the color of an overlapping, semitransparent polygon may be blended with the color of an overlapped polygon according to an opacity level of the overlapping polygon, as shown in the overlapping region of FIG. **1**C. In at least some embodiments, opacity may be stored and indicated as a real number within a range [0.0, 1.0], with 0.0 indicating fully transparent and 1.0 indicating fully opaque. Note that other methods may be used to store and/or indicate opacity, for example integers in the range [0, n] where 0 is fully transparent and n is fully opaque. The opacity of a polygon may be decreased as the polygon grows, so that polygons tend to become more transparent as they grow. As a polygon becomes more transparent (or less opaque), more of the color underneath the polygon is blended with the overlapping polygon. As another example, the wetness of an overlapped polygon may affect the growth of an overlapping polygon. Each polygon may have a wetness attribute. The wetness attribute may be related to the lifespan of the polygon; as long as the polygon is alive, the polygon is considered to be wet. If a second polygon is applied that at least partially overlaps a first polygon that is still wet, the velocity of vertices on the border of the second polygon that are within the first polygon may be increased to account for the wetness of the first polygon. Note that one or more attributes of an overlapped polygon may also be affected by an overlapping polygon. For example, the wetness of the overlapping, live polygon may affect the velocity of at least some of the vertices of an overlapped polygon that is still alive according to its lifespan, and therefore still wet. FIG. **1**C shows examples of overlapped vertices and non-overlapped vertices for the two overlapped polygons. As long as the overlapped vertices of a wet polygon are within another, still wet polygon, the overlapped vertices may tend to grow faster than non-overlapped vertices. These behaviors of overlapped, semitransparent polygons tend to blend two overlapped polygons with a realistic fluid flow effect (see, e.g., FIG. **19**).

Polygons of various sizes, symmetrical or asymmetrical shapes, regular or irregular shapes, and number of vertices on the border(s) may be specified and applied. FIGS. **2**A through **2**D show various examples of polygons that may be used, according to at least some embodiments. FIG. **2**A illustrates an example polygon that has a relatively smooth border, in contrast to the example polygon in FIG. **1**A that has a jagged border. FIG. **2**B illustrates an example polygon that has two borders, an inner border and an outer border, that may be supported in some embodiments. Both the inner border and the outer border include a plurality of vertices, and the vertices on both borders may be grown independently as described herein. The black arrow from an example vertex on the inner border represents a local vector of the inner border, and the black arrow from an example vertex on the outer border represents a local vector of the outer border. Note that, in this example, the outer border tends to expand outward, while the inner border tends to expand inward to fill the "hole." FIGS. **2**C and **2**D illustrate example polygons. In contrast to the substantially circular example polygons of FIGS. **1**A, **2**A, and **2**B, the example polygons of FIGS. **2**C and **2**D have irregular shapes. FIG. **2**C shows a half-moon shaped polygon. Note that the border of the polygon may be defined by multiple vertices, as illustrated by the small circles along the border of this polygon. FIG. **2**D shows another half-moon shaped polygon with jagged edges. A polygon with jagged edges as shown in FIG. **2**D may be defined as an initial shape. However, the polygon shown in FIG. **2**D may also be representative of a polygon after one or more iterations of growing the

border, for example the border of the example polygon of FIG. **2**C. In addition to various sizes, shapes and number of vertices, polygons with various global attribute settings, such as initial direction, velocity, color, opacity, and lifespan, may be specified and applied.

FIG. **3** illustrates global and local attributes of an example polygon, according to at least some embodiments. The border of the polygon is defined by a set of vertices. The vertices may be independently grown from their initial positions when applied to a digital canvas to simulate the spread of a fluid. The polygon may have one or more global attributes, such as a global vector and a lifespan, that may affect the growth of the border. The white arrow in FIG. **3** illustrates an example global vector for the polygon. In at least some embodiments, each vertex on the border of a polygon may have one or more local attributes, such as a local vector, that locally affect the movement of the individual vertex. The black arrows extending from the vertices of the polygon in FIG. **3** illustrate example local vectors. In at least some embodiments, a randomization technique may be applied to the movement at each vertex to simulate the non-uniform expansion along the edges of a deposited fluid, for example in a watercolor simulation.

Note that while FIG. **3** shows an example polygon that includes a global vector that indicates direction (which may influence at least the initial direction of movement of the vertices) and length (which may be taken to represent an initial velocity that may influence at least the initial velocity at which the vertices move), a polygon may have a null or zero global vector. In a polygon with a null global vector, the direction and velocity of the movement of the vertices may be determined primarily by the attributes of the vertices, for example by the local vectors of the vertices. For example, a substantially circular polygon, such as the polygons illustrated in FIGS. **1**A-**1**B, **2**A, and **2**B may have a null global vector so that the vertices tend to expand outward in a more or less uniform fashion according to the vertices' respective local vectors and velocities. The effect of a polygon's global vector on the movement of the vertices will be further discussed below.

Polygons may be applied to a canvas via manual input techniques (e.g., one or more polygons may be applied to a canvas using brush strokes) or via automated techniques (e.g., one or more polygons may be programmatically applied to borders of an object in an input image). Two or more polygons applied to a canvas via one of these techniques may overlap. The polygons may generally be, but are not necessarily, semitransparent. A set of polygons applied by one of these techniques may be, but are not necessarily, of uniform color. FIG. **4** illustrates the application of four instances of the example polygon from FIGS. **1**A and **1**B to a canvas, according to some embodiments. The border of the polygons is shown for clarity. The white arrow indicates the direction of the stroke. Thus, the leftmost polygon was the first applied, and the rightmost polygon was the last applied. Note that each polygon overlaps, or is overlapped by, at least one other polygon.

Groups

As previously mentioned, polygons may be individually applied to the digital canvas, or may be applied in groups that include two or more polygons. A group may be defined as a collection of one or more polygons that may be applied as a unit to a canvas. In a group including two or more polygons, the individual polygons may be of different sizes and shapes, may have a different number of vertices on the border, and may have different global attribute settings, for example different initial directions and/or velocities. Grouping two or more polygons and applying the polygons in groups as a unit may help to achieve various fluid motion effects that are not easy or not possible to achieve by applying a single polygon.

FIGS. **5** through **11** illustrate several example groups that each include two or more polygons, according to some embodiments. Note that, while FIGS. **5**-**7** and **10** show the polygons as circles for illustrative purposes, in practice the borders of each polygon may be represented by vertices connected by edges. The white arrows on the polygons in FIGS. **5** through **10** illustrate example global vectors of the polygons. Note that the global vectors of the various polygons may be different so that the respective polygons have different initial directions and/or different initial velocities

FIG. **5** illustrates an example group that includes three partially overlapping, semitransparent polygons, according to at least some embodiments. Note the region in the center where all three polygons overlap, and that the global vectors are initialized so that the vertices of the three polygons tend to grow outward from the center of the group. However, note that the velocities of the vertices in overlapped portions may be influenced (e.g., sped up) due to the fact that the vertices are in "wet" portions of overlapped or overlapping polygons. This may tend to pull the portions of the border that are in overlapped regions into the other polygons so that the overlapping polygons tend to merge and blend (see, e.g., FIG. **19**).

FIG. **6** illustrates an example group that includes five overlapping polygons, according to at least some embodiments. A center polygon is represented by the bolded circle; the center polygon is surrounded by four outer, partially overlapping polygons. The four outer polygons each has a global vector with direction and velocity; the center polygon has a null or zero global vector.

FIG. **7** illustrates an example group that includes nine overlapping polygons, according to at least some embodiments. A center polygon is represented by the bolded circle; the center polygon is surrounded by eight partially overlapping polygons. Each of the eight outer polygons has a global vector with direction and velocity; the center polygon has a null or zero global vector.

While FIGS. **5** through **7** illustrate example groups that include polygons of substantially the same size and shape, a group may include polygons of different sizes and/or shapes, as illustrated in FIGS. **8** through **11**. FIG. **8** illustrates an example group that includes the example polygon of FIG. **2**A in the center, surrounded by four polygons similar to the example polygon of FIG. **2**C. Each of the four outer polygons partially overlaps the center polygon. The four outer polygons all have global vectors with direction and velocity; the center polygon has a null or zero global vector. FIG. **9** illustrates an example group that includes a substantially hexagonal polygon in the center, surrounded by six irregular polygons similar to the example polygon of FIG. **2**C. The six outer polygons do not initially overlap the center polygon. The six outer polygons all have global vectors with direction and velocity, as shown by the white arrows; the center polygon has a null or zero global vector. Note that the center polygon may have additional vertices between the six shown vertices to provide a more natural fluid motion effect.

While FIGS. **5** through **9** illustrate example groups in which the polygons appear to be arranged in a regular pattern, polygons may be arranged in irregular patterns in a group, as illustrated in FIGS. **10** and **11**. FIG. **10** illustrates an example group that includes a relatively large polygon with three partially overlapping smaller polygons arranged along one side of the larger polygon. Note that, in this example, all four polygons have global vectors with direction and velocity, as shown by the white arrows.

While FIGS. **5** through **10** illustrate example groups in which the polygons partially overlap, a group may include one or more polygons that are completely overlapped by other polygons. For example, FIG. **11** illustrates an example group with two polygons; a larger polygon that completely overlaps or encloses a smaller polygon. Note that, in this example, the smaller polygon has a non-null global vector, while the larger polygon has a null or zero global vector. FIG. **11** may also be used to illustrate the use of water, for example in a watercolor simulation application. The larger polygon in this example may be water, while the smaller, enclosed polygon may be pigment. In at least some embodiments, to designate a polygon as water, the opacity may be set to 0 so that the polygon is fully transparent. A water polygon may, but does not necessarily, grow, but contributes no pigment or shading since it is fully transparent. However, a water polygon may affect other polygons that overlap, or are overlapped by, the polygon. For example, the velocity of the vertices of the smaller, pigment polygon in FIG. **11** may be increased to account for the wetness of the larger, water polygon.

One or more groups each including one or more polygons may be applied as units via manual input techniques (e.g., groups may be applied to a canvas using brush strokes) or via automated techniques (e.g., groups may be programmatically applied to borders of an object in an input image). Two or more groups deposited on a canvas via one of these techniques may overlap. The polygons in the groups may generally be, but are not necessarily, semitransparent. The polygons in each group applied by one of these techniques may be, but are not necessarily, of uniform color. FIG. **12** illustrates the application of eight instances of the example group from FIG. **6** to a canvas, according to at least some embodiments. The borders of the polygons are not shown. The black arrow indicates the direction of the stroke that applied the groups. Thus, the leftmost polygon was the first applied, and the rightmost polygon was the last applied. Note that each polygon overlaps, or is overlapped by, at least one other polygon, and that each group overlaps, or is overlapped by, at least one other group.

Vertex, Polygon and Group Attributes

In at least some embodiments, each vertex of a polygon may have a set of one or more attributes, such as local vector, position, and velocity, that may be specifically assigned to that vertex. In addition, each polygon may have a set of one or more attributes, such as a global vector, color, opacity, and velocity, that may be specifically assigned to the polygon. Two polygons in a group may have different values for at least some attributes, such as a global vector, velocity and opacity. In addition, a group may have a set of one or more attributes that may be applied to polygons within the group. For example, a group color attribute and a group lifespan attribute may be applied to, or inherited by, all of the polygons in the group. Other examples include, but are not limited to, opacity, which in some embodiments may be set at the group level and inherited by or applied to polygons within the group.

The following illustrates, at an abstract level, how a data structure for representing a group of one or more polygons may be formed in at least some embodiments, and is not intended to be limiting. Note that, in at least some embodiments, the polygons may be stored in a format, or alternatively converted to a format, that can be processed by a graphics processing unit (GPU):

```
GROUP X
    <group attributes, e.g. lifespan, size, opacity, velocity color, etc.>
    Group Polygon List:
        POLYGON A
            <polygon attributes, e.g. global vector, velocity, etc.>
            Polygon Vertex List:
                VERTEX 1
                    <vertex attributes, e.g. local vector, velocity,
                    position>
                VERTEX 2
                    <vertex attributes, e.g. local vector, velocity,
                    position >
                .
                .
                .
            <end Polygon Vertex List >
        POLYGON B
            <polygon attributes, e.g. global vector, velocity, etc.>
            Polygon Vertex List:
                VERTEX 1
                    <vertex attributes, e.g. local vector, velocity,
                    position >
                VERTEX 2
                    <vertex attributes, e.g. local vector, velocity,
                    position >
                .
                .
                .
            <end Polygon Vertex List >
        .
        .
        .
    <end Group Polygon List>
<end GROUP X>
```

Brushes

In at least some embodiments, various brushes (or more generally applicators) may be defined; a defined brush may then be used to apply particular groups of one or more polygons as units to a digital canvas. Multiple units, generally but not necessarily overlapping, may be applied by a brush in one action, for example via a stroke applied using a cursor control device such as a mouse. For example, a graphical user interface may be provided via which a user may select a brush and apply strokes to a canvas using the selected brush, each stroke applying one or more units to the canvas according to factors including the path, length, and speed of the stroke. As another example, a graphical user interface may be provided via which a user may select a brush that may be used to programmatically apply groups of polygons to borders of an object in an input image. A brush may have various attributes, such as size, opacity, and color, that may be applied to or inherited by the units of the brush. In some embodiments, a brush may be defined with preset attributes, such as size and opacity. In some embodiments, a user may change one or more attributes of a brush, for example the color to be applied by the brush, prior to or between strokes. For example, in some embodiments, a user may set or change the color of a brush before a stroke is applied by selecting a color (or a blending of colors) from a palette user interface. As another example, in a watercolor simulation application or similar application, a user may add water to a brush before a stroke is applied to dilute the color of the brush and also to change the fluid dynamics (e.g., the velocity) of the watercolor paint.

The following illustrates, at an abstract level, how a data structure for representing a brush may be formed in at least some embodiments, and is not intended to be limiting:

```
BRUSH Z
    <brush attributes, e.g. size, opacity, velocity, color, lifespan, etc.>
    GROUP X (inherits one or more attributes from the brush)
<end BRUSH Z>
```

While this example shows a brush Z that includes one group X that may be applied using the brush, in at least some embodiments a brush may include and apply two or more different groups.

FIG. **13** illustrates a set of example brush strokes made by example brushes that may be defined, for example for a watercolor simulation application, according to at least some embodiments. FIG. **13** shows 29 different brushes (brushes **0** through **28**), each brush including at least one group that may be applied to a canvas using the respective brush. In addition, some of the brushes (e.g., brushes **6**A-B, **11**A-B, and **12**A-C) have different brush attributes assigned to similar groups, so that there are two number **6**, two number **11**, and three number **12** brushes. Brush **6**, for example, has two versions: brush **6**A, size **5**, and brush **6**B, size **10**. As another example, brush **12** has three versions: brush **12**A, size **5**, opacity **5**; brush **12**B, size **10**, opacity **5**; and brush **12**C, size **10**, opacity **10**. Also note that brushes **21** through **28** are similar, but differ primarily in direction or vector.

FIGS. **14** through **16** show some of the example brushes from FIG. **13** in more detail. FIG. **14** shows example brush **1** from FIG. **13** as including a group with one substantially circular polygon. The black arrows show the direction of growth of the vertices of the polygon; the vertices tend to grow outward from the center of the polygon. FIG. **15** shows example brush **13** from FIG. **13** as including a group with multiple substantially circular polygons. Each polygon in the group includes a global vector that defines or influences the growth of the respective polygon.

In at least some embodiments, different polygons in a group may have different attributes that affect the visual results of painting with the brush. For example, in some embodiments, different polygons in a group may represent different substances, such as water and pigment, that may be used to generate various painting effects. As an example, FIG. **16** shows example brush **15** from FIG. **13** as including a group with multiple substantially circular polygons, where one of the polygons is water and the others are pigment (e.g., watercolor pigment). The larger outer polygon contains water, while the three smaller, inner polygons arranged on one side of the larger polygon contain paint or pigment. In at least some embodiments, to designate a polygon as water, the opacity may be set to 0 so that the polygon is fully transparent. The inner, paint polygons have global vectors that tend to spread the paint towards the near edge of the larger, water polygon. The larger polygon has a null or zero global vector, so the water tends to spread outward relatively evenly in all directions according to the local vectors of the vertices. Note that, since the inner, paint polygons are within the outer, water polygon, the wetness of the outer polygon affects the velocity of the vertices of the inner polygons. Applying this group with a brush may have an effect as shown in brush **15**, where a "leading edge" of the brush stroke has a higher concentration of pigment, while the trailing edge has less pigment.

Polygon Application, Growth, and Blending

In at least some embodiments, polygons may be applied to a canvas in groups of one or more polygons, as previously described. Polygons may be applied to a canvas via manual input techniques (e.g., one or more polygons may be applied to a canvas using brush strokes) or via automated techniques (e.g., one or more polygons may be programmatically applied, for example to borders of an object in an input image). Thus, polygon application may be event-driven or programmatic. In either case, a fluid motion simulation module may include an applicator submodule that enables and performs the application of the polygons to the canvas.

The growth and blending of the applied polygons may be performed by a process executing in the background. For example, a background process may be scheduled to execute 30 times a second, or 60 times a second, or at some other periodic or aperiodic interval, with a growth and blending step and possibly a rendering of the resulting image performed at each execution. The displayed image may be, but is not necessarily, updated at each execution of the background process. The background process may execute according to its cycle or schedule even if there are no user interface events (e.g., mouse-down events) or programmatic events between executions. Each polygon may include a lifespan attribute that may be initialized when the polygon is initially deposited on the canvas. At each execution, the background process may grow each polygon that is still alive according to the polygon's lifespan. The lifespan may, for example, be indicated by a counter that is initialized to some value and that is decremented (or, alternatively, incremented) at each execution of the background process; the polygon is alive until the counter reaches a specified threshold, e.g. zero. Alternatively, the lifespan may be time-based; that is, the polygon may be initialized to be alive for two seconds, or some other period. When that time is elapsed, the polygon is no longer alive, and thus is not grown by the background process. Alternatively, the growth and blending of the applied polygons may be performed by a process implemented as a programmatic loop, and thus not as a background process.

In at least some embodiments, at least part of the growth and blending process, for example the rendering of the updated polygons, may be performed on a GPU or GPUs. FIG. **27** shows an example fluid motion simulation module **920** that includes an applicator submodule **926** and a growth and blending submodule **928**; the growth and blending submodule represents the background process or programmatic loop that grows, blends and renders the polygons. An example computer system on which embodiments of the fluid motion simulation module **920** may be implemented is illustrated in FIG. **28**. FIGS. **21** and **22**A-**22**C illustrate example user interfaces to a fluid motion simulation module that may be used in at least some embodiments.

In at least some embodiments, each deposited polygon is grown by independently moving the vertices of the polygon according to local vectors of the vertices; one or more other attributes of the vertex, polygon, and/or group may affect the movement. In at least some embodiments, each vertex has a position (e.g., an (x,y) coordinate) and a local vector or direction. A velocity, or speed, attribute affects how far a vertex may move at each movement. In at least some embodiments, a randomization technique may be applied to the movement at each vertex to simulate the non-uniform expansion along the edges of a deposited fluid. In at least some embodiments, at an initialization step for a polygon, at each vertex the global vector of the polygon and the initial local vector of the vertex are summed to generate a starting local vector for the vertex. Thus, the global vector of the polygon, if not null or zero, may influence the initial direction of movement at each vertex. In at least some embodiments, after the initialization of the polygon, at each growth step of the polygon (i.e., at each execution of the growth and blending submodule), at each vertex of the polygon, the following equation may be used to compute the new position to which the vertex is to be moved:

$$P_{new}=P_{old}+V*S+R$$

where P_{old} designates the current position of the vertex, P_{new} designates the new position for the vertex, V designates the local vector for the vertex, S designates the speed or velocity at the vertex, and R represents randomized values that may be added to the (x,y) coordinates of the vertex; the randomized values may be positive or negative.

In at least some embodiment, S (the speed, or velocity) at each vertex may be obtained from or influenced by the velocity attribute of the respective polygon. That is, a velocity attribute of the polygon may determine or affect the velocity of the vertices. In some embodiments, S may be obtained from or influenced by the velocity attribute of the group to which the polygon belongs. In at least some embodiments, S may be adjusted according to the lifespan attribute of the polygon. For example, the speed of the polygon may be decreased with respect to the age of the polygon, so that the vertices tend to move faster when the polygon is first deposited, and tend to slow down as the polygon ages.

In at least some embodiments, at each vertex of a polygon, before computing P_{new} for the vertex, a check may be performed to determine if the vertex is on a wet polygon that is overlapped by the vertex's polygon. If the vertex is on a wet polygon, then S (the speed, or velocity) at the vertex may be increased to account for the wetness of the overlapped polygon.

In at least some embodiments, to blend a polygon with an object beneath the polygon (e.g., the canvas, or another, overlapped polygon), a blending technique may be applied in which the color of the polygon is multiplied by an alpha value and added to the color of the object underneath the polygon, multiplied by (1−alpha). The blending equation that may be used in at least some embodiments may be expressed as:

Output color=alpha*polygon color+(1−alpha)*color of object to be blended.

The colors may be, but are not necessarily, RGB colors. Alpha may be expressed as a floating-point number, for example in the range [0,1]. In at least some embodiments, alpha may be, or may be derived from, the current opacity level of the polygon. In at least some embodiments, the shading of the polygon tends to get lighter or darker as the polygon grows. In at least some embodiments, to achieve this effect, the opacity of the polygon is decreased (or the transparency is increased) as the polygon increases in area. The color of the polygon remains constant, but the color is blended with the color of the object beneath the polygon, which tends to lighten or darken the shading depending on the respective colors. Note that other techniques for blending may be used in some embodiments.

FIGS. **17** through **19** illustrate polygon growth and blending, according to at least some embodiments. These Figures are given for illustrative purposes, and are not intended to be limiting. Each frame in these Figures illustrates an example of an image that may displayed. The arrows between the frames represent at least one execution of a background process or programmatic loop that performs growth, blending, and rendering of the polygon(s).

FIG. **17** illustrates the growth and blending of a polygon, similar to the example polygon of FIGS. **1**A and **1**B, deposited on a digital canvas, according to at least some embodiments. FIG. **17**, frame **0** shows the polygon as initially deposited on the canvas. FIG. **17**, frames **1** through **8** illustrate the growth of the polygon by independently moving the vertices of the polygon in a background process or programmatic loop, as previously described. Note that, in this example, the shading of the polygon tends to get lighter as the polygon grows. In at least some embodiments, to achieve this effect, the opacity of the polygon is decreased (or the transparency is increased) as the polygon increases in area. The color of the polygon remains constant, but the color is blended with the color of the object(s) beneath the polygon, in this example the color of the canvas. Since, in this example, the canvas is lighter than the color of the polygon, as the opacity decreases, the shading of the polygon thus becomes lighter.

FIG. **18** illustrates the growth and blending of a polygon, similar to the example polygon of FIGS. **1**A and **1**B, deposited on an existing painted region, for example a region of previously deposited, but still wet paint, or alternatively a region of water that has been deposited on the canvas, according to at least some embodiments. The existing region is represented by the relatively lightly shaded area within the dashed line. FIG. **18**, frame **0** shows the polygon as initially deposited on the existing region. FIG. **18**, frames **1** through **8** illustrate the growth of the polygon by independently moving the vertices of the polygon in a background process or programmatic loop, as previously described. As previously noted, the wetness of an overlapped polygon may be accounted for by increasing the velocity of the vertices of an overlapping polygon. In this example, this effect causes the deposited polygon to rapidly spread out into the existing wet region. However, when a portion of the border of the polygon reaches the edge of the existing wet region, as can be seen beginning at frame **2**, the vertices along this portion of the border of the polygon tend to slow down, as the vertices no longer lie on the wet overlapped polygon.

Since, in this example, the shading of the existing region is lighter than the color of the polygon, as the opacity decreases, the shading of the polygon thus becomes lighter as more of the color of the existing region overlapped by the polygon comes through.

The example shown in FIG. **18** may be used to illustrate a painting effect, for example a watercolor effect, that embodiments may enable. The existing region bounded by the dashed line may represent a shape deposited as a template. The template may be composed of one or more polygons, which may be referred to as template polygons. The attributes of each template polygon may be set so that the polygon is alive and thus wet, but with a velocity of 0, a null global vector, and null local vectors so that the polygon does not grow, but still affects other polygons that overlap the template polygon. A pigmented polygon deposited on the shape formed by the template polygon(s) may tend to grow outward to conform to the shape). Template polygons may be used, for example, to simulate a watercolor effect that is used in real watercolor painting in which a painter deposits paint in water previously deposited on a canvas; the deposited paint then spreads out into the water. To simulate this effect, one or more template polygons may be deposited with an opacity of 0 to represent water, a velocity of 0, and null vectors. The user may then deposit paint polygon(s) in the water; the deposited paint polygon(s) then expand to fill the shape formed by the water template polygon(s).

FIG. **19** illustrates the deposition, growth and blending of two partially overlapping polygons similar to the example polygon of FIGS. **1**A and **1**B, according to at least some embodiments. FIG. **19**, frame **0** shows two polygons A and B as initially deposited on a canvas. The borders of the polygons are shown for illustrative purposes. The overlapping region of the two polygons is indicated. FIG. **19**, frames **1** through **9** illustrate the growth of the polygons by independently moving the vertices of both polygons in a background process or programmatic loop, as previously described. As previously noted, the wetness of overlapping polygons may be accounted for by increasing the velocity of the vertices of one

or both polygons. In this example, both polygons are wet, so this effect causes portions of each deposited polygon that lie within the other polygon to spread out into the other polygon. However, portions of the borders of the polygons that do not lie within the other polygon tend to grow slower, as the vertices do not lie on the other, wet polygon. Frame **9** shows a border for the overlapping region as a dashed line; note that the overlapping region has expanded into both polygons, and has grown much faster than the non-overlapped portions of the polygons.

Note that the shading of the overlapping region is a blending of the colors of the two polygons. While this image is in grayscale, if the images were color images, the blended colors of the polygon would be appropriately blended as a color mix. For example, if polygon A was blue and polygon B was yellow, the overlapping region would generally exhibit some shade of green, depending on the values of the blue and yellow polygons and on the opacity level. Also note that, as the polygons grow, the opacity of the polygons decreases; as the opacity decreases, the shading of the polygons, including the shading of overlapped regions, may become lighter or more diluted.

While FIG. **19** shows two overlapping polygons, more than two polygons may overlap, as shown in FIGS. **4** and **12**. The motion and blending effects may be applied to the overlapping regions, which may include portions of more than two polygons. For example, the velocity of more than one overlapped wet polygon may be used to alter the velocity of at least some of the vertices of an overlapping polygon. As another example, more than two polygons may be blended. In some embodiments, this blending of multiple polygons may be performed by first blending the overlapping portions of the two bottommost polygons, and then blending the results with the next highest polygon, and so on

Watercolor Simulation

As previously noted, an example application for embodiments of the vector-based, fluid motion simulation technique described herein is in watercolor simulation. For example, embodiments of the polygons, groups, brushes, and polygon growth and blending techniques as described herein may be used in a watercolor simulation application. A user interface may be provided via which a user may apply watercolor paint, water, and mixtures of paint and water to canvas using different brushes to simulate natural watercolor painting. FIG. **20**A is an image that shows several example real strokes and effects achieved via natural watercolor painting. FIG. **20**B illustrates each of the real examples in FIG. **20**A simulated with an embodiment of the vector-based, fluid motion simulation technique as described herein. FIGS. **21** and **22**A-**22**C illustrate example user interfaces that may be used in at least some embodiments of a watercolor simulation application that employs the vector-based, fluid motion simulation technique described herein.

Example User Interface

FIGS. **21** and **22**A-**22**C illustrate example user interfaces that may be used with at least some embodiments of the vector-based, fluid motion simulation technique, for example in watercolor simulation applications. These user interfaces are given by way of example, and are not intended to be limiting.

FIG. **21** illustrates a user interface that may be used with at least some embodiments of the vector-based, fluid motion simulation technique, for example in a watercolor simulation application. The user interface **200** may provide a canvas **210** area on which painting is performed via one or more user interface elements and techniques. The user interface **200** may provide a menus **202** area that includes one or more menu user interface elements via which a user may select various menu items to perform various functions or select various tools of the application. For example, a menu may include an open option via which a user can open a new canvas **210**, a save option via which a user may save a current canvas **210**, and a print option via which a user may print a current canvas **210**.

The user interface **200** may also provide a brush palette **204** that includes one or more user interface elements via which a user may select brushes for depositing paint on the canvas **210**. The user interface **200** may also provide a color palette **206** that includes one or more user interface elements via which a user may select (or blend) pigments for paint to be deposited on the canvas **210** with a brush. While not shown, in some embodiments, the user interface **200** may also provide a user interface element via which a user may select water to be applied by a brush or add water to dilute the pigment(s) on a brush.

The user interface **200** may also include one or more other user interface elements or controls **208**, which may include but are not limited to buttons, slider bars, check boxes, radio buttons, text entry boxes, and popup menus, via which a user may, for example, set or modify one or more attributes of the vector-based, fluid motion simulation technique as described herein, for example attributes of a polygon, a group, or a brush. For example, the user interface **200** may provide a control via which a user may set or change the opacity for paint to be applied by a brush. As another example, the user interface **200** may provide a control via which a user may set or change the size or diameter of strokes to be applied by a brush.

FIGS. **22**A through **22**C illustrate several methods that may be used to interact with the user interface **200** to deposit paint on the canvas **210** and to perform other actions, according to at least some embodiments of the vector-based, fluid motion simulation technique. FIG. **22**A illustrates a method in which the user uses a cursor control device **230**, for example a mouse, to control a brush cursor **240** displayed on a display device **220**. To apply a stroke, the user may move the cursor control device **230** to cause a corresponding movement of the brush cursor **240** on the user interface **200**. The black arrows indicate a brush stroke motion of the cursor control device **230**, and a corresponding stroke applied to the canvas **210** with the cursor **240**. Note that, in some embodiments, the user may, for example, hold down a mouse button when applying a brush stroke, releasing the mouse button when the stroke is done. The user may use the cursor **240** to perform other functions on the user interface, for example to select from the brush palette **204**, color palette **206**, and other controls **208**.

In at least some embodiments, a touch method, where a user interacts with the user interface **200** via a touch-enabled device using gestures applied with one or more digits and/or a stylus, may be used. The touch-enabled device may be a conventional touch device in which a single digit or stylus is used, or a multi-touch device that accepts and processes substantially simultaneous input from two or more digits. FIG. **22**B illustrates a method that employs a touch-enabled device **250**, such as a touch-sensitive pad, to apply strokes to a canvas **210** displayed on a display device **220**. The black arrows indicate a brush stroke gesture applied via the touch-enabled device **250**, and a corresponding stroke applied to the canvas **210**. FIG. **22**C illustrates a method that employs a touch-enabled display device **260** to apply strokes to a canvas **210** displayed on the display device **260**. The black arrow indicates a brush stroke gesture applied via the touch-enabled display device **260**.

Some touch-enabled devices may be pressure-sensitive, and thus may support pressure functions. For example, a touch pad as illustrated in FIG. **22**B may be pressure-sensitive. With such a device, at least some embodiments may support various functions enabled by pressure applied via a stylus to the touch surface. For example, varying levels of pressure applied by a stylus to the touch surface may be used to define the size of the brush, the opacity of the polygons or groups, the number of vertices in polygons, and so on. In addition, the angle or tilt of the stylus may be used to control the shape of the brush, the type of the brush, the direction (global vector) of the brush, or other brush attributes.

FIG. **23** illustrates an example of watercolor painting using the vector-based, fluid motion simulation technique, according to at least some embodiments. The watercolor painting illustrated in FIG. **23** may be performed for example, using a user interface as illustrated in FIGS. **21** and **22**A-**22**C. FIG. **23** is given for illustrative purposes, and is not intended to be limiting. Each frame in FIG. **23** illustrates an example of an image that may displayed. The arrows between the frames represent at least one execution of a background process or programmatic loop that performs growth, blending, and rendering of the polygon(s). The arrows that appear on some of the frames point to strokes that have been applied, but do not indicate the stroke motion itself FIG. **23**, frame **0** shows a stroke as initially deposited on the canvas. FIG. **23**, frames **1** through **7** illustrate the depositing of additional strokes of various paints using various brushes, and the growth and blending of the deposited paint by independently moving the vertices of the polygons composing the deposited paint and blending of overlapping polygons with overlapped polygons in a background process or programmatic loop, as previously described. FIG. **23**, frame **1** points out a second stroke applied to the canvas. FIG. **23**, frame **2** points out an area in which one or more additional strokes have been applied. FIG. **23**, frame **3** points out a new, overlapping stroke that has been applied. FIG. **23**, frames **4**, **5**, and **6** show several additional strokes and the growth and blending of the applied strokes. FIG. **23**, frame **7** shows a resulting image after the strokes applied on the previous frames have spread and blended, and the lifespans of the polygons applied by the strokes on the previous frames has ended (i.e., the paint has "dried.")

FIG. **24** is a high-level flowchart of a painting method that employs the vector-based, fluid motion simulation technique, according to at least some embodiments. As indicated at **300**, simulated fluid, for example simulated watercolor paint, may be deposited on a canvas as one or more groups, each group including one or more polygons. The polygons may be, but are not necessarily, semitransparent. For example, paint may be deposited to a canvas by a user using strokes applied using a brush according to a user interface method as illustrated in FIGS. **21**, **22**A-**22**C, and **23**. As another example, paint may be deposited to a canvas programmatically.

In at least some embodiments, the polygons deposited at **300** may be processed by a method implemented in a growth and blending process **310**. For example, growth and blending process **310** may be implemented as a background process that may be scheduled to execute 30 times a second, 60 times a second, or at some other periodic or aperiodic interval. Element **312** may occur at each execution of the background process; elements **314** and **316** may only be performed if there is at least one live polygon.

At **312** of the growth and blending process **310**, the method may check to see if there are any live polygons according to the lifespan attribute of the polygons. If there are live polygons at **312**, then the method may grow the live polygons by independent movement of the vertices of the polygons, as indicated at **314**. In at least some embodiments, a randomization technique may be applied to the movement at each vertex of each polygon. The method may then render and display the polygons on the canvas, as indicated at **316**. Rendering and displaying the polygons may include blending overlapping semitransparent polygons with overlapped polygons or other overlapped objects, such as the canvas itself. At **312** of the growth and blending process **310**, if there are no live polygons, then the background process may wait for a next execution.

At **302**, the method may return to element **300** to deposit additional simulated fluid to the canvas, if not done depositing fluid. For example, a user may choose to add additional paint to a canvas via the user interface. Additional polygons deposited at **300** may be processed by the background process **310**. At **302**, if done depositing paint, note that the growth and blending process **310** may continue to process the deposited polygons until there are no more live polygons, that is until the paint is "dry."

The above describes the growth and blending process **310** as being implemented as a background process. Alternatively, the growth and blending process may be implemented as a loop in an event-driven program. In this implementation, the event-driven program may check for events that indicate the deposition of group(s) (e.g., mouse events that indicate a stroke applied to the canvas). When a group deposition event is detected, one or more iterations of elements **312** through **316** of the growth and blending process **310** may be performed. When the growth and blending process **310** is done, the program waits for a next event.

FIG. **25** is a flowchart of a method for processing a deposited polygon, according to at least some embodiments of the vector-based, fluid motion simulation technique. As indicated at **400**, a polygon may be deposited to a canvas. The deposited polygon may, but does not necessarily, overlap another polygon. As indicated at **402**, at an initialization step, for each vertex of the polygon, a local vector may be determined from the global vector of the polygon and the initial local vector of the vertex. In at least some embodiments, the global vector of the polygon and the initial local vector of the vertex are summed to generate the starting local vector for the vertex.

In at least some embodiments, the polygon deposited at **400** may be processed by a method implemented in a growth and blending process **410**. For example, growth and blending process **410** may be implemented as a background process **410** that may be scheduled to execute 30 times a second, 60 times a second, or at some other periodic or aperiodic interval. Element **412** may occur at each execution of the background process **410**; elements **414** through **418** may only be performed if the polygon is alive.

At **412** of the growth and blending process, the method may check to see if the polygon is alive according to the lifespan attribute of the polygon. If the polygon is alive, then the method may grow the polygon by independent movement of the vertices of the polygon. As indicated at **414**, for each vertex of the polygon, a new position for the vertex may be determined from the current position of the polygon, the local vector of the polygon, one or more attributes, and a randomization factor. In at least some embodiments, at each vertex, the following equation may be used to compute the new position to which the vertex is to be moved:

$$P_{new}=P_{old}+V*S+R$$

where P_{old} designates the current position of the vertex, P_{new} designates the new position for the vertex, V designates the local vector for the vertex, S designates the speed or velocity at the vertex, and R represents randomized values that may be

added to the (x,y) coordinates of the vertex; the randomized values may be positive or negative. In at least some embodiments, at each vertex of a polygon, before computing P_{new} for the vertex, a check may be performed to determine if the vertex is on a wet polygon that is overlapped by the vertex's polygon. If the vertex is on a wet polygon, then S (the speed, or velocity) at the vertex may be increased to account for the wetness of the overlapped polygon.

As indicated at **416**, the polygon may be blended with overlapped polygons, if any. In at least some embodiments, to blend a polygon with an overlapped polygon, a blending technique may be applied in which the color of the polygon is multiplied by an alpha value and added to the color of the overlapped polygon, multiplied by (1−alpha). The blending equation that may be used in at least some embodiments may be expressed as:

Output color=alpha*overlapping polygon color+(1−alpha)*overlapped polygon color.

In at least some embodiments, alpha may be a floating-point number, for example in the range [0,1], and may be derived from, the current opacity level of the overlapping polygon.

As indicated at **410**, the polygon may be aged. The lifespan of the polygon may, for example, be indicated by a counter that is initialized to some value and that is decremented (or, alternatively, incremented) at each execution of the growth and blending process **410**; the polygon is alive until the counter reaches a specified threshold, e.g. zero. Alternatively, the lifespan may be time-based; that is, the polygon may be initialized to be alive for two seconds, or some other period. When that time is elapsed, the polygon is no longer alive, and thus is no longer grown by the background process.

The above describes the growth and blending process **410** as being implemented as a background process. Alternatively, the growth and blending process may be implemented as a loop in an event-driven program. In this implementation, the event-driven program may check for events that indicate the deposition of polygon(s) (e.g., mouse events that indicate a stroke applied to the canvas). When a polygon deposition event is detected, element **402** may be performed, followed by one or more iterations of elements **412** through **418** of the growth and blending process **410**. When the growth and blending process **410** is done, the program waits for a next event.

FIG. **26** is a flowchart of a method for moving a vertex of a polygon, according to at least some embodiments of the vector-based, fluid motion simulation technique. As indicated at **450**, a check is made to determine if the vertex is within a wet polygon. At **452**, if the vertex is within a wet polygon as determined at **450**, the velocity of the vertex may be increased to account for the wetness of the overlapped polygon. Otherwise, the velocity is not increased. At **456**, a new position for the vertex may be calculated by multiplying the local vector of the vertex by the velocity at the vertex and adding the resulting offset to the previous position of the vertex. As indicated at **458**, in at least some embodiments, a randomization factor may be added to the new position to simulate the random motion of a fluid at the edges.

Some embodiments may include a means for performing the vector-based, fluid motion simulation technique. For example, a fluid motion simulation module may receive input specifying at least a portion of a digital canvas on which one or more polygons as described herein have been deposited, and may grown and blend the polygons to simulate the dynamics of a fluid such as paint, as described herein. The fluid motion simulation module may in some embodiments be implemented by a non-transitory, computer-readable storage medium and one or more processors (e.g., CPUs and/or GPUs) of a computing apparatus. The computer-readable storage medium may store program instructions executable by the one or more processors to cause the computing apparatus to perform receiving input specifying at least a portion of a digital canvas on which one or more polygons as described herein have been deposited, and may grow and blend the polygons to simulate the dynamics of a fluid such as paint, as described herein. Other embodiments of the fluid motion simulation module may be at least partially implemented by hardware circuitry and/or firmware stored, for example, in a non-volatile memory.

Example Implementations

FIG. **27** illustrates a fluid motion simulation module that may implement embodiments of the vector-based, fluid motion simulation techniques illustrated in FIGS. **1** through **26**. Module **920** may include an applicator submodule **926** that provides a painting technique and that employs a user interface **924** similar to the example user interface illustrated in FIGS. **21** and **22**A-**22**C to deposit polygons to a canvas **910**. Module **920** may also include a growth and blending submodule **928** that implements a fluid motion simulation technique that grows and blends deposited polygons as illustrated in FIGS. **24** through **26**. FIG. **28** illustrates an example computer system on which embodiments of module **920** may be implemented. Module **920** may receive as input a digital canvas **910**. While FIG. **27** shows the digital canvas **910** as blank, the canvas **910** may be a digital image or photograph, and may be an image that was previously created with module **920**. Applicator submodule **926** may receive user input **912**, for example brush strokes, via user interface **924** depositing polygons to working canvas **922**. Growth and blending submodule **928**, which may execute as a background process, may grow and blend the deposited polygons, for example as illustrated in FIGS. **24** through **26**. Module **920** generates as output an output image **930**. Output image **930** may, for example, be stored to a storage medium **940**, such as system memory, a disk drive, DVD, CD, etc., displayed to a display device **950**, printed on a printer device (not shown), and/or passed to one or more other module(s) **960** for additional processing.

In some embodiments, fluid motion simulation module **920** may provide a user interface **924** via which a user may interact with the module **920**, for example to specify or select brushes, to select a pigment and/or water for application by a selected brush, to set one or more attributes for polygons or groups to be deposited, and to perform a painting method as described herein. In some embodiments, the user interface may provide user interface elements whereby the user may select attribute values for groups or polygons to be deposited including one or more of, but not limited to, opacity, speed, and diameter.

Example System

Embodiments of a fluid motion simulation module and/or of the various vector-based, fluid motion simulation techniques as described herein may be executed on one or more computer systems, which may interact with various other devices. One such computer system is illustrated by FIG. **28**. In different embodiments, computer system **1000** may be any of various types of devices, including, but not limited to, a personal computer system, desktop computer, laptop, notebook, or netbook computer, mainframe computer system, handheld computer, workstation, network computer, a camera, a set top box, a mobile device, a consumer device, video game console, handheld video game device, application server, storage device, a peripheral device such as a switch, modem, router, or in general any type of computing or electronic device.

In the illustrated embodiment, computer system **1000** includes one or more processors **1010***a*, **1010***b*, . . . **1010***n* coupled to a system memory **1020** via an input/output (I/O) interface **1030**. Computer system **1000** further includes a network interface **1040** coupled to I/O interface **1030**, and one or more input/output devices **1050**, such as cursor control device **1060**, keyboard **1070**, and display(s) **1080***l* In some embodiments, it is contemplated that embodiments may be implemented using a single instance of computer system **1000**, while in other embodiments multiple such systems, or multiple nodes making up computer system **1000**, may be configured to host different portions or instances of embodiments. For example, in one embodiment some elements may be implemented via one or more nodes of computer system **1000** that are distinct from those nodes implementing other elements.

In various embodiments, computer system **1000** may be a uniprocessor system including one processor **1010**, or a multiprocessor system including several processors **1010** (e.g., two, four, eight, or another suitable number). Processors **1010** may be any suitable processor capable of executing instructions. For example, in various embodiments, processors **1010** may be general-purpose or embedded processors implementing any of a variety of instruction set architectures (ISAs), such as the x86, PowerPC, SPARC, or MIPS ISAs, or any other suitable ISA. In multiprocessor systems, each of processors **1010** may commonly, but not necessarily, implement the same ISA.

In some embodiments, at least one processor **1010** may be a graphics processing unit. A graphics processing unit or GPU may be considered a dedicated graphics-rendering device for a personal computer, workstation, game console or other computing or electronic device. Modern GPUs may be very efficient at manipulating and displaying computer graphics, and their highly parallel structure may make them more effective than typical CPUs for a range of complex graphical algorithms. For example, a graphics processor may implement a number of graphics primitive operations in a way that makes executing them much faster than drawing directly to the screen with a host central processing unit (CPU). In various embodiments, the vector-based, fluid motion simulation techniques disclosed herein may, at least in part, be implemented by program instructions configured for execution on one of, or parallel execution on two or more of, such GPUs. The GPU(s) may implement one or more application programmer interfaces (APIs) that permit programmers to invoke the functionality of the GPU(s). Suitable GPUs may be commercially available from vendors such as NVIDIA Corporation, ATI Technologies (AMD), and others.

System memory **1020** may be configured to store program instructions and/or data accessible by processor **1010**. In various embodiments, system memory **1020** may be implemented using any suitable memory technology, such as static random access memory (SRAM), synchronous dynamic RAM (SDRAM), nonvolatile/Flash-type memory, or any other type of memory. In the illustrated embodiment, program instructions and data implementing desired functions, such as those described above for embodiments of a fluid motion simulation module, are shown stored within system memory **1020** as program instructions **1025** and data storage **1035**, respectively. In other embodiments, program instructions and/or data may be received, sent or stored upon different types of computer-accessible media or on similar media separate from system memory **1020** or computer system **1000**. Generally speaking, a computer-accessible medium may include storage media or memory media such as magnetic or optical media, e.g., disk or CD/DVD-ROM coupled to computer system **1000** via I/O interface **1030**. Program instructions and data stored via a computer-accessible medium may be transmitted by transmission media or signals such as electrical, electromagnetic, or digital signals, which may be conveyed via a communication medium such as a network and/or a wireless link, such as may be implemented via network interface **1040**.

In one embodiment, I/O interface **1030** may be configured to coordinate I/O traffic between processor **1010**, system memory **1020**, and any peripheral devices in the device, including network interface **1040** or other peripheral interfaces, such as input/output devices **1050**. In some embodiments, I/O interface **1030** may perform any necessary protocol, timing or other data transformations to convert data signals from one component (e.g., system memory **1020**) into a format suitable for use by another component (e.g., processor **1010**). In some embodiments, I/O interface **1030** may include support for devices attached through various types of peripheral buses, such as a variant of the Peripheral Component Interconnect (PCI) bus standard or the Universal Serial Bus (USB) standard, for example. In some embodiments, the function of I/O interface **1030** may be split into two or more separate components, such as a north bridge and a south bridge, for example. In addition, in some embodiments some or all of the functionality of I/O interface **1030**, such as an interface to system memory **1020**, may be incorporated directly into processor **1010**.

Network interface **1040** may be configured to allow data to be exchanged between computer system **1000** and other devices attached to a network, such as other computer systems, or between nodes of computer system **1000**. In various embodiments, network interface **1040** may support communication via wired or wireless general data networks, such as any suitable type of Ethernet network, for example; via telecommunications/telephony networks such as analog voice networks or digital fiber communications networks; via storage area networks such as Fibre Channel SANs, or via any other suitable type of network and/or protocol.

Input/output devices **1050** may, in some embodiments, include one or more display terminals, keyboards, keypads, touchpads, scanning devices, voice or optical recognition devices, or any other devices suitable for entering or retrieving data by one or more computer system **1000**. Multiple input/output devices **1050** may be present in computer system **1000** or may be distributed on various nodes of computer system **1000**. In some embodiments, similar input/output devices may be separate from computer system **1000** and may interact with one or more nodes of computer system **1000** through a wired or wireless connection, such as over network interface **1040**.

As shown in FIG. **28**, memory **1020** may include program instructions **1025**, configured to implement embodiments of a fluid motion simulation module as described herein, and data storage **1035**, comprising various data accessible by program instructions **1025**. In one embodiment, program instructions **1025** may include software elements of embodiments of a fluid motion simulation module as illustrated in the above Figures. Data storage **1035** may include data that may be used in embodiments. In other embodiments, other or different software elements and data may be included.

Those skilled in the art will appreciate that computer system **1000** is merely illustrative and is not intended to limit the scope of a fluid motion simulation module as described herein. In particular, the computer system and devices may include any combination of hardware or software that can perform the indicated functions, including a computer, personal computer system, desktop computer, laptop, notebook,

or netbook computer, mainframe computer system, handheld computer, workstation, network computer, a camera, a set top box, a mobile device, network device, internet appliance, PDA, wireless phones, pagers, a consumer device, video game console, handheld video game device, application server, storage device, a peripheral device such as a switch, modem, router, or in general any type of computing or electronic device. Computer system **1000** may also be connected to other devices that are not illustrated, or instead may operate as a stand-alone system. In addition, the functionality provided by the illustrated components may in some embodiments be combined in fewer components or distributed in additional components. Similarly, in some embodiments, the functionality of some of the illustrated components may not be provided and/or other additional functionality may be available.

Those skilled in the art will also appreciate that, while various items are illustrated as being stored in memory or on storage while being used, these items or portions of them may be transferred between memory and other storage devices for purposes of memory management and data integrity. Alternatively, in other embodiments some or all of the software components may execute in memory on another device and communicate with the illustrated computer system via inter-computer communication. Some or all of the system components or data structures may also be stored (e.g., as instructions or structured data) on a computer-accessible medium or a portable article to be read by an appropriate drive, various examples of which are described above. In some embodiments, instructions stored on a computer-accessible medium separate from computer system **1000** may be transmitted to computer system **1000** via transmission media or signals such as electrical, electromagnetic, or digital signals, conveyed via a communication medium such as a network and/or a wireless link. Various embodiments may further include receiving, sending or storing instructions and/or data implemented in accordance with the foregoing description upon a computer-accessible medium. Accordingly, the present invention may be practiced with other computer system configurations.

CONCLUSION

Various embodiments may further include receiving, sending or storing instructions and/or data implemented in accordance with the foregoing description upon a computer-accessible medium. Generally speaking, a computer-accessible medium may include storage media or memory media such as magnetic or optical media, e.g., disk or DVD/CD-ROM, volatile or non-volatile media such as RAM (e.g. SDRAM, DDR, RDRAM, SRAM, etc.), ROM, etc., as well as transmission media or signals such as electrical, electromagnetic, or digital signals, conveyed via a communication medium such as network and/or a wireless link.

The various methods as illustrated in the Figures and described herein represent example embodiments of methods. The methods may be implemented in software, hardware, or a combination thereof. The order of method may be changed, and various elements may be added, reordered, combined, omitted, modified, etc.

Various modifications and changes may be made as would be obvious to a person skilled in the art having the benefit of this disclosure. It is intended that the invention embrace all such modifications and changes and, accordingly, the above description to be regarded in an illustrative rather than a restrictive sense.

What is claimed is:

1. A method for simulating motion of a fluid deposited on a digital canvas, comprising:

depositing, by a computing device, a plurality of polygons representing the fluid on the digital canvas, a border of each deposited polygon being defined by a plurality of vertices, each vertex including a local vector that indicates direction at the vertex, each deposited polygon including a global vector that indicates a direction of the polygon;

for each deposited polygon, prior to growing a border of the polygon, adjusting the local vector at each vertex of the polygon according to the global vector of the respective polygon; and

for each deposited polygon, iteratively growing the border of the respective polygon, the iteratively growing the border of a polygon comprising, at each iteration, independently moving each vertex of the respective polygon according to a local vector at the vertex and a velocity at the vertex.

2. The method as recited in claim **1**, wherein said independently moving each vertex of the respective polygon according to a local vector at the vertex and a velocity at the vertex comprises calculating a new position for the vertex according to the local vector at the vertex and the velocity at the vertex and applying a randomization to coordinates of the new position.

3. The method as recited in claim **1**, wherein each polygon includes a lifespan attribute that specifies a period for which the respective polygon is alive after said depositing the respective polygon, wherein a polygon that is alive is considered wet and a polygon that is not alive is considered dry, and wherein said growing the border is performed only on polygons that are wet according to their respective lifespan attribute.

4. The method as recited in claim **3**, wherein said independently moving each vertex of the respective polygon according to a local vector at the vertex and a velocity at the vertex comprises, at each vertex, determining if the vertex is within another polygon that is wet according to its lifespan attribute and, if the vertex is within another polygon that is wet, increasing the velocity at the vertex to account for the wetness of the other polygon.

5. The method as recited in claim **1**, wherein at least one of the plurality of deposited polygons at least partially overlaps at least one other deposited polygon, the method further comprising, for each of the plurality of polygons that at least partially overlaps at least one other polygon, blending overlapping regions of the polygon with overlapped regions of the at least one other polygon according to an opacity level of the overlapping polygon.

6. The method as recited in claim **1**, wherein iteratively growing the border of a polygon further comprises decreasing an opacity level of the polygon to account for growth of the polygon.

7. The method as recited in claim **1**, wherein said depositing a plurality of polygons comprises depositing the polygons in a plurality of groups each including two or more of the plurality of polygons, and wherein at least two of the two or more polygons in each group differ according to at least one of shape, size, position, velocity, and global vector.

8. The method as recited in claim **1**, wherein said depositing a plurality of polygons representing the fluid on the digital canvas comprises receiving input via a user interface indicating one or more strokes applied to the digital canvas via a brush, wherein at least one of the plurality of polygons is

deposited along each of the one or more strokes according to direction and velocity of the respective stroke.

9. The method as recited in claim **8**, wherein the brush specifies one or more brush attributes that are applied to polygons deposited by the brush, wherein the brush attributes include at least one of size, opacity, velocity, color, and lifespan, and wherein at least one of the brush attributes can be changed via the user interface prior to or after each stroke.

10. The method as recited in claim **9**, wherein the user interface provides a plurality of different selectable brushes, wherein each brush deposits polygons in groups each including one or more polygons, and wherein the groups deposited by at least two of the brushes differ according to at least one of number, shape, and arrangement of polygons within the groups.

11. A system, comprising:

at least one processor; and

a memory comprising program instructions, the program instructions being executable by the at least one processor to implement a fluid motion simulation module operable to:

obtain input indicating deposition of a plurality of polygons representing a fluid on a digital canvas, a border of each deposited polygon being defined by a plurality of vertices, each vertex including a local vector that indicates direction at the vertex; and

iteratively grow the border of each deposited polygon, by configuring the fluid motion simulation module to be operable to, at each iteration, independently move each vertex of the respective polygon according to a local vector at the vertex and a velocity at the vertex, and to calculate a new position for the vertex according to the local vector at the vertex and the velocity at the vertex and apply a randomization to coordinates of the new position.

12. The system as recited in claim **11**, wherein each polygon includes a lifespan attribute that specifies a period for which the respective polygon is alive after said depositing the respective polygon, wherein a polygon that is alive is considered wet and a polygon that is not alive is considered dry, and wherein the fluid motion simulation module is operable to grow the border only of polygons that are wet according to their respective lifespan attribute.

13. The system as recited in claim **12**, wherein at least one of the plurality of deposited polygons at least partially overlaps at least one other deposited polygon, and wherein, to independently move each vertex of the respective polygon according to a local vector at the vertex and a velocity at the vertex, the fluid motion simulation module is operable to, at each vertex, determine if the vertex is within another polygon that is wet according to its lifespan attribute and, if the vertex is within another polygon that is wet, increase the velocity at the vertex to account for the wetness of the other polygon.

14. The system as recited in claim **11**, wherein said input indicating the deposition of the plurality of polygons is obtained via one or more brush strokes applied to the digital canvas via a user interface to the fluid motion simulation module, wherein the brush strokes are applied with a brush that deposits polygons in groups each including one or more polygons, and wherein at least one of the plurality of polygons is deposited along each of the one or more strokes according to direction and velocity of the respective stroke.

15. A non-transitory computer-readable storage medium comprising program instructions stored thereon, the program instructions being computer-executable to implement a fluid motion simulation module operable to:

obtain input indicating deposition of a plurality of polygons representing a fluid on a digital canvas, a border of each deposited polygon being defined by a plurality of vertices, each vertex including a local vector that indicates direction at the vertex, and at least one of the plurality of deposited polygons at least partially overlapping at least one other deposited polygon, each deposited polygon including a lifespan attribute that specifies a period for which the respective polygon is alive after said depositing the respective polygon, a deposited polygon that is alive being considered wet and a deposited polygon that is not alive being considered dry; and

iteratively grow the border of each deposited polygon by configuring the fluid motion simulation module is to be operable to, at each iteration, independently move each vertex of the respective polygon according to a local vector at the vertex and a velocity at the vertex, said fluid motion simulation module being operable to grow the border of deposited polygons that are wet according to their respective lifespan attribute.

16. The non-transitory computer-readable storage medium as recited in claim **15**, wherein to independently move each vertex of the respective polygon according to a local vector at the vertex and a velocity at the vertex, the fluid motion simulation module is operable to calculate a new position for the vertex according to the local vector at the vertex and the velocity at the vertex and apply a randomization to coordinates of the new position.

17. The non-transitory computer-readable storage medium as recited in claim **15**, wherein, to independently move each vertex of the respective polygon according to a local vector at the vertex and a velocity at the vertex, the fluid motion simulation module is operable to, at each vertex, determine if the vertex is within another polygon that is wet according to its lifespan attribute and, if the vertex is within another polygon that is wet, increase the velocity at the vertex to account for the wetness of the other polygon.

18. The non-transitory computer-readable storage medium as recited in claim **15**, wherein said input indicating the deposition of the plurality of polygons is obtained via one or more brush strokes applied to the digital canvas via a user interface to the fluid motion simulation module, wherein the brush strokes are applied with a brush that deposits polygons in groups each including one or more polygons, and wherein at least one of the plurality of polygons is deposited along each of the one or more strokes according to direction and velocity of the respective stroke.

19. A method for simulating motion of a fluid deposited on a digital canvas, comprising:

depositing a plurality of polygons, by a computing device, representing the fluid on the digital canvas, at least one of the plurality of deposited polygons at least partially overlaps at least one other deposited polygon, a border of each deposited polygon being defined by a plurality of vertices, each vertex including a local vector that indicates direction at the vertex;

for each of the plurality of polygons that at least partially overlaps at least one other polygon, blending overlapping regions of the polygon with overlapped regions of the at least one other polygon according to an opacity level of the overlapping polygon; and

for each deposited polygon, iteratively growing the border of the respective polygon, the iteratively growing the border of a polygon comprising, at each iteration, inde-

pendently moving each vertex of the respective polygon according to a local vector at the vertex and a velocity at the vertex.

20. A method for simulating motion of a fluid deposited on a digital canvas, comprising:

depositing a plurality of polygons, by a computing device, representing the fluid on the digital canvas, a border of each deposited polygon being defined by a plurality of vertices, each vertex including a local vector that indicates direction at the vertex; and

for each deposited polygon, iteratively growing the border of the respective polygon, the iteratively growing the border of a polygon comprising decreasing an opacity level of the polygon to account for growth of the polygon, and, at each iteration, independently moving each vertex of the respective polygon according to a local vector at the vertex and a velocity at the vertex.

21. A method embodying program instructions executable by one or more processors for simulating motion of a fluid deposited on a digital canvas, comprising:

depositing a plurality of polygons, by a computing device, representing the fluid on the digital canvas, a border of each deposited polygon being defined by a plurality of vertices, each vertex including a local vector that indicates direction at the vertex, said depositing a plurality of polygons comprising depositing the polygons in a plurality of groups each including two or more of the plurality of polygons, at least two of the two or more polygons in each group differing according to at least one of shape, size, position, velocity, and global vector; and

for each deposited polygon, iteratively growing the border of the respective polygon, the iteratively growing the border of a polygon comprising, at each iteration, independently moving each vertex of the respective polygon according to a local vector at the vertex and a velocity at the vertex.

* * * * *